# Programação Inteira e Inteira Mista no Planejamento de Lavra

**Ebook didático de Pesquisa Operacional aplicada à mineração.** Este material ensina a transformar decisões de lavra (quantos caminhões, quais frentes abrir, quais blocos lavrar em cada período) em modelos de Programação Inteira (PI) e Programação Inteira Mista (PIM), e mostra como o método Branch and Bound encontra e prova a solução ótima.




**CONHECER. PLANEJAR. TRANSFORMAR.**

EBOOK DIDÁTICO • ENGENHARIA DE MINAS • 2026

# Sumário

Apresentação ..... 3

01 Por que variáveis inteiras? ..... 4

02 Modelando com variáveis binárias e inteiras ..... 6

03 Relaxamento linear e os perigos do arredondamento ..... 10

04 Branch and Bound para Programação Inteira ..... 13

05 Branch and Bound para Programação Inteira Mista ..... 18

06 Formulações fortes, cortes e solvers ..... 23

07 PI e PIM no planejamento de lavra ..... 26

08 Resolvendo no computador com Python e PuLP ..... 31

09 Exercícios propostos ..... 36

Questionário de revisão ..... 38

Glossário ..... 40

Referências ..... 42

## Apresentação

Este ebook foi escrito para estudantes de Engenharia de Minas que já conhecem Programação Linear (PL) e estão tendo o primeiro contato com variáveis inteiras. Ele acompanha as aulas 4.1 (Programação Inteira) e 4.2 (Programação Inteira Mista) da disciplina e se apoia no capítulo 11 de Hillier e Lieberman e em artigos de planejamento de lavra.

Todos os exemplos usam números pequenos, para que as contas possam ser refeitas à mão. Os valores de margem, custo e capacidade são didáticos: eles preservam a estrutura dos problemas reais, mas não representam uma mina específica.

### Como o ebook está organizado

| Capítulo | Pergunta que responde |
|---|---|
| 1. Por que variáveis inteiras? | Quando a PL deixa de servir e que tipos de modelo existem |
| 2. Modelando com binárias e inteiras | Como escrever regras de operação como restrições lineares |
| 3. Relaxamento e arredondamento | Por que arredondar a solução da PL não funciona |
| 4. Branch and Bound para PI | Como o método divide, limita e poda |
| 5. Branch and Bound para PIM | O que muda quando há variáveis contínuas |
| 6. Formulações fortes e cortes | Por que dois modelos corretos podem ter desempenhos tão diferentes |
| 7. PI e PIM no planejamento de lavra | Como esses modelos aparecem na prática e na pesquisa |
| 8. Resolvendo no computador | Como montar e resolver os modelos em Python |
| 9. Exercícios propostos | Como praticar e conferir o aprendizado |

### Como estudar

1. Leia cada exemplo com lápis e papel e refaça as contas.
2. Antes de ler a solução, tente escrever o modelo sozinho.
3. Em toda variável binária, faça o "teste de mesa": substitua 0 e 1 e veja se a restrição faz sentido na mina.
4. Use a lista **Exercícios Resolvidos — Programação Inteira Mista no Planejamento de Mina** como companheira deste ebook. Ela traz oito exercícios com árvores desenhadas.

### Notação usada

| Símbolo | Significado |
|---|---|
| x, x1, x2... | Variáveis de decisão (contínuas ou inteiras, conforme o modelo) |
| y | Variável binária: y = 1 se a decisão acontece; y = 0 caso contrário |
| n | Número inteiro de equipamentos |
| Z | Valor da função objetivo |
| Z* | Valor da titular (melhor solução inteira encontrada até o momento) |
| SP0, SP1... | Subproblemas (nós) da árvore de Branch and Bound |
| M | Constante "grande" usada em restrições lógicas |
| ⌊x⌋ | Parte inteira de x (arredondar para baixo): ⌊2,67⌋ = 2 |

# Capítulo 1 — Por que variáveis inteiras?

**Objetivos do capítulo:** reconhecer decisões que não podem ser fracionárias; distinguir variáveis contínuas, inteiras e binárias; classificar um modelo como PL, PI pura, PI binária ou PIM.

## 1.1 Quando a Programação Linear deixa de servir

Na Programação Linear, toda variável pode assumir qualquer valor real. Isso funciona muito bem para quantidades divisíveis: toneladas lavradas, horas de operação, proporção de cada pilha numa mistura. Lavrar 3,7 kt numa frente faz todo o sentido.

Mas muitas decisões de uma mina não admitem frações:

- Não existe comprar 2,67 caminhões ou alocar 4,5 escavadeiras.
- Não existe "abrir um terço" de uma frente nem instalar "meio" britador.
- Se o bloco é a unidade de decisão do sequenciamento, não se lavra 40% dele num ano e 60% no outro.

Quando a resposta precisa ser um número inteiro, ou uma decisão do tipo sim/não, acrescentamos ao modelo a **condição de integralidade**. O objetivo e as restrições continuam lineares; o que muda é o conjunto de valores que as variáveis podem assumir.

## 1.2 Três tipos de variável

| Tipo | O que representa | Exemplos na mineração |
|---|---|---|
| Contínua | Quantidade divisível | Toneladas, horas, teor de uma mistura |
| Inteira | Contagem indivisível | Caminhões, escavadeiras, equipes, turnos |
| Binária (0 ou 1) | Decisão sim/não | Abrir frente, lavrar bloco, comprar britador, aprovar projeto |

A variável binária é uma inteira especial, limitada a 0 e 1. Ela é a peça mais importante deste ebook: com ela, regras lógicas ("se abrir, então paga"; "ou A ou B"; "B só depois de A") viram restrições lineares.

## 1.3 As quatro partes de uma formulação

1. **Variáveis de decisão:** o que o planejador pode escolher, cada uma com significado e unidade.
2. **Função objetivo:** o que se quer maximizar ou minimizar (lucro, VPL, produção, custo).
3. **Restrições:** os limites da operação (orçamento, capacidade da usina, frota, oficina, precedência, qualidade).
4. **Domínio:** quais variáveis são contínuas, inteiras ou binárias.

Na PL, o domínio quase sempre se resume a "$x \geq 0$". Na PI e na PIM, o domínio passa a ser parte essencial do modelo.

## 1.4 Classificação dos modelos

| Modelo | Domínio das variáveis | Exemplo na mineração |
|---|---|---|
| PL | Todas contínuas | Blendagem de minério, em kt/dia |
| PI pura | Todas inteiras | Número de caminhões de cada porte |
| PI binária (PIB) | Todas em {0, 1} | Carteira de projetos; lavrar ou não cada bloco |
| PI mista (PIM) | Algumas inteiras/binárias, outras contínuas | Abrir frentes (binária) e quanto lavrar em cada uma (contínua) |

Na literatura em inglês, a PIM aparece como **MIP** (Mixed Integer Programming) ou **MILP** (Mixed Integer Linear Programming). Os artigos de planejamento de lavra citados no Capítulo 7 usam essas siglas.

O modelo geral da PIM é:

$$\max \; Z = \sum_{j=1}^{n} c_j \, x_j$$

$$\text{s.a.:} \quad \sum_{j=1}^{n} a_{ij} \, x_j \leq b_i \qquad (i = 1, \ldots, m)$$

$$x_j \geq 0 \qquad (j = 1, \ldots, n)$$

$$x_j \text{ inteira para } j = 1, \ldots, I \quad (I \leq n)$$

Se I = n, todas as variáveis são inteiras (PI pura). Se I = 0, nenhuma é (PL). Entre esses extremos está a PIM.

## 1.5 Por que isso é difícil?

À primeira vista, a PI parece mais fácil que a PL: há "menos" soluções possíveis. Na prática, é o contrário.

- Na PL, o ótimo está sempre num vértice da região viável, e o método simplex caminha de vértice em vértice com eficiência.
- Na PI, as soluções são pontos isolados. Não há um "caminho" entre eles, e o número de combinações cresce de forma explosiva: com n variáveis binárias, existem $2^n$ combinações.

| Número de binárias | Combinações ($2^n$) | Exemplo |
|---|---|---|
| 3 | 8 | Abrir ou não 3 frentes |
| 10 | 1.024 | Carteira com 10 projetos |
| 30 | mais de 1 bilhão | 30 frentes candidatas |
| 100.000 | número com mais de 30 mil dígitos | 10 mil blocos × 10 períodos |

Testar todas as combinações (enumeração completa) só é possível em problemas pequenos. Problemas como o sequenciamento de lavra pertencem à classe dos problemas **NP-difíceis**: não se conhece algoritmo que os resolva exatamente em tempo polinomial. O Branch and Bound (Capítulos 4 e 5) contorna isso com uma **enumeração implícita**: ele descarta grupos inteiros de combinações sem precisar testá-las uma a uma.

> ***Resumo do capítulo:*** *use variáveis contínuas para quantidades divisíveis, inteiras para contagens e binárias para decisões sim/não. A PIM combina os três tipos e é o modelo mais comum no planejamento de lavra.*

# Capítulo 2 — Modelando com variáveis binárias e inteiras

**Objetivos do capítulo:** traduzir regras lógicas em restrições lineares; ligar variáveis binárias a variáveis contínuas; fazer o teste de mesa de cada restrição.

O "truque" da PI e da PIM está em usar variáveis binárias para ligar ou desligar partes do modelo. Este capítulo reúne as ferramentas mais usadas no planejamento de lavra. Para cada uma, há a regra em palavras, a formulação e o teste de mesa.

## 2.1 Regras lógicas entre decisões sim/não

Sejam x1, x2, ... variáveis binárias (xj = 1 se o projeto j for aprovado).

| Regra em palavras | Formulação | Exemplo na mina |
|---|---|---|
| No máximo um entre dois | x1 + x3 ≤ 1 | Britador in-pit ou nova frota de caminhões, nunca os dois |
| Exatamente um entre N | x1 + x2 + ... + xN = 1 | Escolher um único modelo de britador |
| Pelo menos k entre N | x1 + ... + xN ≥ k | Abrir pelo menos 2 frentes |
| Dependência (2 só se 1) | x2 ≤ x1 | Correia transportadora só se houver britador in-pit |
| Ambos ou nenhum | x1 = x2 | Pilha de homogeneização e sua retomadora |

Teste de mesa da dependência x2 ≤ x1:

| x1 | x2 | x2 ≤ x1? | Significado |
|---|---|---|---|
| 0 | 0 | sim | Nenhum dos dois |
| 1 | 0 | sim | Só o britador |
| 1 | 1 | sim | Britador e correia |
| 0 | 1 | **não** | Correia sem britador: proibido |

### Exemplo 2.1 — Carteira de projetos da mina

Com R$ 10 milhões para investir, a mina avalia quatro projetos. A correia depende do britador, e o britador é excludente com a nova frota (o orçamento de engenharia só comporta uma dessas duas obras).

| Projeto | VPL (R$ mi) | Custo (R$ mi) |
|---|---|---|
| 1. Britador in-pit | 9 | 6 |
| 2. Correia transportadora | 5 | 3 |
| 3. Nova frota | 6 | 5 |
| 4. Ampliação da pilha | 4 | 2 |

$$\max \; Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$$
$$\text{s.a.:} \quad 6x_1 + 3x_2 + 5x_3 + 2x_4 \le 10 \quad \text{(orçamento)}$$
$$x_1 + x_3 \le 1 \quad \text{(britador ou frota)}$$
$$x_2 \le x_1 \quad \text{(correia depende do britador)}$$
$$x_j \in \{0,1\}$$

| Carteira | Custo | VPL | Situação |
|---|---|---|---|
| {1, 2} | 9 | **14** | Viável e ótima |
| {1, 4} | 8 | 13 | Viável |
| {3, 4} | 7 | 10 | Viável |

| Carteira | Custo | VPL | Situação |
|---|---|---|---|
| {1, 2, 4} | 11 | 18 | Excede o orçamento |
| {2, 4} | 5 | 9 | Viola a dependência |

A carteira ótima é aprovar o britador e a correia: custo de R$ 9 milhões e VPL de R$ 14 milhões. Este exemplo volta no Capítulo 6, para mostrar como um plano de corte fortalece o modelo.

## 2.2 O kit de modelagem da PIM

Na PIM, a binária controla uma variável contínua. As seis ferramentas abaixo cobrem a maioria das regras de operação.

### Ferramenta 1 — Custo fixo

Abrir uma frente custa F (acesso, drenagem, mobilização) mais c por tonelada. O custo é 0 se x = 0 e F + c·x se x > 0. Esse "salto" em x = 0 não é linear.

$$\texttt{custo} = F\,y + c\,x, \qquad x \le M\,y, \qquad y \in \{0,1\}$$

| y | x ≤ M·y vira | Custo |
|---|---|---|
| 0 | x ≤ 0 (frente fechada) | 0 |
| 1 | x ≤ M (lavra até a capacidade) | F + c·x |

Escolha M = capacidade da frente. Assim, a mesma restrição liga x a y e impõe a capacidade.

### Ferramenta 2 — Produção mínima

Regra: "ou a frente não opera, ou opera com pelo menos L (lote mínimo), até a capacidade U".

$$L\,y \le x \le U\,y$$

Com y = 0, x = 0. Com y = 1, L ≤ x ≤ U. A produção positiva abaixo de L fica proibida.

### Ferramenta 3 — Restrições ou-ou

A usina opera com a configuração de peneiras A **ou** B, e cada uma impõe um limite diferente. Somamos M ao lado direito da restrição que deve ser "desligada":

$$f_A(x) \le b_A + M\,y$$
$$f_B(x) \le b_B + M\,(1-y)$$

Com y = 0, vale A e B fica desligada. Com y = 1, vale B e A fica desligada. "Desligar" significa que a restrição nunca limita: M deve ser o maior valor que f(x) pode atingir menos b.

### Ferramenta 4 — K de N restrições

Um contrato aceita a carga se pelo menos 2 das 3 especificações (Fe, $SiO_2$, P) forem atendidas.

$$g_i(x) \le b_i + M\,y_i \quad (i = 1,2,3), \qquad y_1 + y_2 + y_3 \le 1$$

yi = 1 "libera" a especificação i. Regra geral: para que K de N restrições valham, no máximo N − K podem ser liberadas: Σ yi ≤ N − K.

### Ferramenta 5 — Escolha entre N valores

Três modelos de britador (500, 800 e 1.200 t/h; R$ 2,0, 3,0 e 4,5 milhões). Só um será comprado:

$$\texttt{Capacidade} = 500y_1 + 800y_2 + 1200y_3, \quad \texttt{Custo} = 2{,}0y_1 + 3{,}0y_2 + 4{,}5y_3, \quad y_1 + y_2 + y_3 = 1$$

A capacidade escolhida entra nas demais restrições, por exemplo: produção ≤ horas × capacidade.

### Ferramenta 6 — Equipamento × produção

Equipamentos são inteiros, mas a produção que geram é contínua. A ligação é feita pela produtividade p de cada unidade:

$$x \leq p\, n, \qquad n \in \{0, 1, 2, \dots\}$$

Com equipamentos em série (escavadeira → caminhões → britador), cada elo gera uma restrição, e a produção fica limitada pelo gargalo.

### Resumo do kit

| Regra da operação | Formulação | Variáveis |
|---|---|---|
| Custo fixo de abertura | F·y + c·x; x ≤ M·y | y binária |
| Produção mínima | L·y ≤ x ≤ U·y | y binária |
| Uma ou outra restrição | f1 ≤ b1 + M·y; f2 ≤ b2 + M(1 − y) | y binária |
| K de N restrições | fi ≤ bi + M·yi; Σ yi ≤ N − K | yi binárias |
| Escolher 1 de N valores | Σ vi·yi; Σ yi = 1 | yi binárias |
| Equipamento × produção | x ≤ p·n | n inteira |

## 2.3 Precedência e sequenciamento de blocos

No planejamento a céu aberto, um bloco só pode ser lavrado depois dos blocos que estão acima dele (definidos pelo ângulo de talude). Seja xbt = 1 se o bloco b for lavrado no período t.

$$\begin{aligned}
\max \quad & \sum_{b}\sum_{t} VPL_{bt}\, x_{bt} \\
\text{s.a.:} \quad & \sum_{t} x_{bt} \leq 1 && \text{(cada bloco é lavrado no máximo uma vez)} \\
& \sum_{b} m_b\, x_{bt} \leq C_t && \text{(capacidade de lavra no período } t\text{)} \\
& x_{bt} \leq \sum_{\tau \leq t} x_{a\tau} && \text{(precedência: } a \text{ está acima de } b\text{)} \\
& x_{bt} \in \{0,1\}
\end{aligned}$$

A restrição de precedência diz: para lavrar b no período t, o bloco a precisa ter sido lavrado no mesmo período ou antes. Com um único período, ela se reduz a xb ≤ xa, a mesma dependência da seção 2.1. VPLbt já inclui o desconto: o mesmo bloco vale menos se for lavrado mais tarde.

Esse é o modelo clássico de sequenciamento de lavra. Em depósitos reais ele tem milhões de variáveis e de arcos de precedência, e por isso é o centro do Capítulo 7.

## 2.4 Boas práticas de modelagem

1. Descreva a decisão em palavras antes de escrever equações.
2. Escreva o significado de cada binária: "y = 1 se...; y = 0 caso contrário".
3. Confira as unidades de cada termo de cada restrição.
4. Faça o teste de mesa com y = 0 e y = 1 em cada restrição de ligação.
5. Use o menor M válido, de preferência uma capacidade física (Capítulo 6).
6. Não crie binárias desnecessárias: cada uma pode dobrar o tamanho da árvore.
7. Mantenha contínuas as variáveis que podem ser contínuas.
8. Depois de resolver, confira se a solução faz sentido operacional.

> ***Resumo do capítulo:*** *binárias transformam lógica em restrições lineares. Na PIM, a ligação entre binárias e contínuas (x ≤ M·y, L·y ≤ x ≤ U·y, x ≤ p·n) é o coração do modelo, e o teste de mesa é a forma de verificar se ela está certa.*

# Capítulo 3 — Relaxamento linear e os perigos do arredondamento

**Objetivos do capítulo:** definir o relaxamento linear; entender por que ele fornece um limite; ver, num gráfico, por que arredondar a solução da PL pode dar uma resposta inviável ou ruim.

## 3.1 O relaxamento linear

O **relaxamento linear** de um modelo de PI ou PIM é o mesmo modelo sem as condições de integralidade. Uma variável inteira passa a aceitar qualquer valor real; uma binária passa a aceitar qualquer valor entre 0 e 1.

Como o relaxamento contém todas as soluções inteiras viáveis e ainda outras (as fracionárias), o seu ótimo nunca é pior que o ótimo inteiro:

$$\text{Maximização:}\quad Z_{\text{PL}} \geq Z_{\text{PI}} \qquad\qquad \text{Minimização:}\quad Z_{\text{PL}} \leq Z_{\text{PI}}$$

Essa propriedade é a base do Branch and Bound. Ela permite três conclusões práticas:

1. **Limite:** numa maximização, Z do relaxamento é um limite superior para o melhor que se pode conseguir com valores inteiros.
2. **Atalho:** se a solução do relaxamento já sair inteira, ela é ótima para o problema inteiro.
3. **Inviabilidade:** se o relaxamento for inviável, o problema inteiro também é.

## 3.2 Exemplo 3.1 — Compra de frota

Uma mina pode comprar caminhões grandes e médios e quer maximizar a capacidade de transporte. O orçamento é de R$ 53 milhões e a oficina suporta 26 horas de manutenção por dia.

| Por caminhão | Grande (x1) | Médio (x2) | Limite |
|---|---|---|---|
| Capacidade (kt/dia) | 9 | 5 | maximizar |
| Custo (R$ milhões) | 8 | 5 | 53 |
| Manutenção (h/dia) | 5 | 2 | 26 |

$$\begin{aligned} &\max Z = 9x_1 + 5x_2 \quad (\text{kt/dia}) \\ \text{s.a.:}\quad &8x_1 + 5x_2 \leq 53 \quad (\text{orçamento}) \\ &5x_1 + 2x_2 \leq 26 \quad (\text{oficina}) \\ &x_1, x_2 \geq 0 \text{ e inteiras} \end{aligned}$$

**Relaxamento.** O ótimo da PL está no encontro das duas retas. Da oficina, x2 = (26 − 5x1)/2. Substituindo no orçamento: 8x1 + 5(26 − 5x1)/2 = 53, o que dá x1 = 8/3 ≈ 2,67 e x2 = 19/3 ≈ 6,33, com Z = 167/3 ≈ 55,67 kt/dia. Nenhuma frota real transporta mais do que isso.

## 3.3 Por que arredondar não funciona

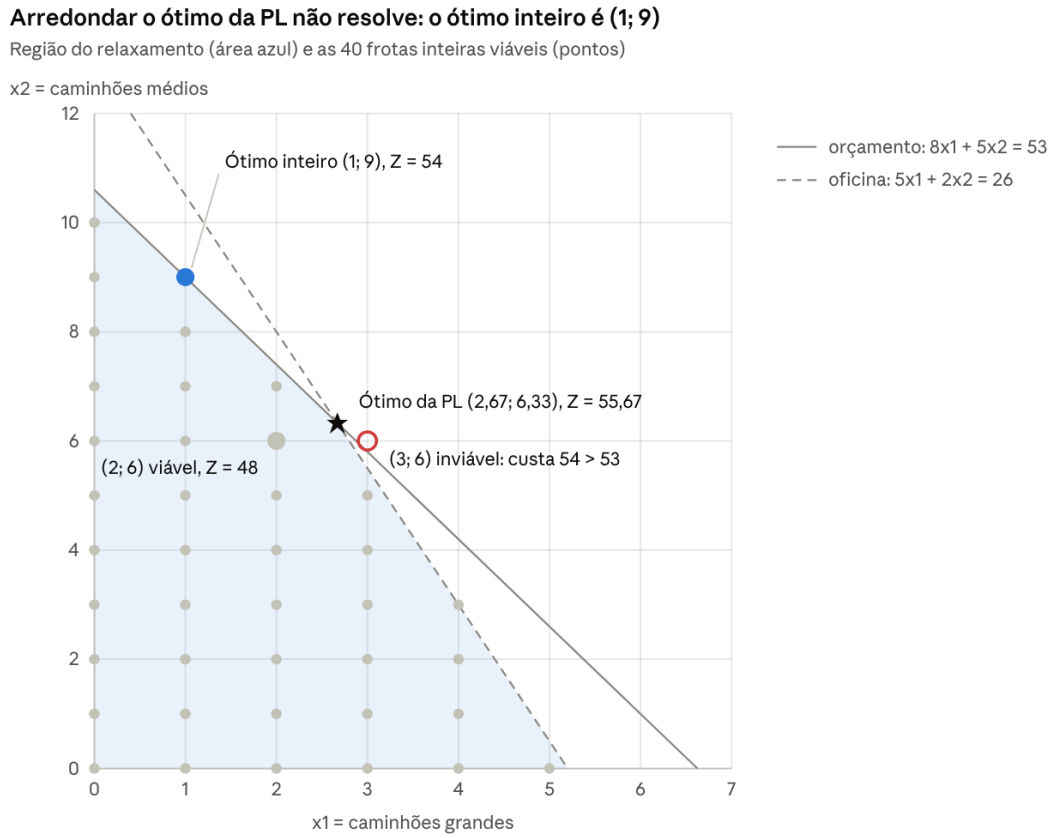



Figura 3.1 · Exemplo 3.1 (frota), calculado para este ebook

| Tentativa | Frota (x1; x2) | Capacidade (kt/dia) | Checagem |
|---|---|---|---|
| Arredondar ao mais próximo | (3; 6) | — | **Inviável:** custa 8·3 + 5·6 = 54 > 53 |
| Arredondar para baixo | (2; 6) | 48 | Viável, mas sobram R$ 7 milhões |
| Ótimo inteiro | (1; 9) | **54** | Custa exatamente 53; oficina 23 h |

O ótimo inteiro (1; 9) está longe do ótimo da PL: não é nenhum dos quatro vizinhos de (2,67; 6,33). Existem 40 frotas inteiras viáveis neste problema, e nada no relaxamento aponta diretamente para a melhor delas. Arredondar pode:

- dar uma solução **inviável** (3; 6);
- dar uma solução viável, mas **pior** que a ótima (2; 6), aqui com 6 kt/dia a menos (11%);
- não dar **nenhuma garantia** de qualidade, mesmo quando acerta.

Quando os valores são grandes (por exemplo, 1.837,4 toneladas), arredondar costuma ser aceitável, porque a diferença é desprezível. O problema aparece quando os valores são pequenos (número de caminhões) ou binários (abrir ou não): arredondar y = 0,5 para 0 ou 1 muda completamente a decisão.

## 3.4 O gap de integralidade

A diferença entre o ótimo do relaxamento e o ótimo inteiro chama-se **gap de integralidade**. No Exemplo 3.1, ele vale 55,67 − 54 = 1,67 kt/dia, ou 3,1% do ótimo.

Quanto menor o gap, mais "apertado" é o relaxamento e mais fácil o Branch and Bound trabalha. Dois modelos corretos do mesmo problema podem ter gaps muito diferentes; o Capítulo 6 mostra como reduzi-lo.

Nos relatórios dos solvers aparece um conceito parecido, o **gap de otimalidade**, medido durante a busca:

$$\text{gap} = \frac{\text{melhor limite} - Z^*}{Z^*}$$

Gap zero significa que o solver provou a otimalidade. Gap de 2% significa que a solução atual está, no máximo, 2% abaixo do ótimo, mesmo sem conhecermos o ótimo.

## 3.5 Arredondar o limite (e não a solução)

Há um arredondamento que é válido. Se todas as variáveis forem inteiras e todos os coeficientes da função objetivo forem inteiros, então Z de qualquer solução inteira também é inteiro. Nesse caso, o limite 55,67 pode ser arredondado para baixo: nenhuma frota inteira transporta mais do que **55** kt/dia. Hillier e Lieberman usam essa regra no algoritmo para PI pura, e ela é usada no Capítulo 4.

Na PIM essa regra **não vale** em geral: as variáveis contínuas podem fazer Z ser fracionário, e o limite deve ser usado como está. A exceção é quando Z assume valores inteiros em todas as soluções viáveis (por exemplo, se as variáveis contínuas não entram na função objetivo).

## 3.6 Quando o relaxamento já sai inteiro

Alguns problemas têm uma estrutura especial (matriz de restrições totalmente unimodular e lado direito inteiro) que faz todos os vértices do relaxamento serem inteiros; assim, o simplex já encontra uma solução ótima inteira. Sem o lado direito inteiro, essa garantia não vale. É o caso dos problemas de transporte, de designação e de fluxo em redes. Na mineração, o exemplo mais importante é o **limite final de cava** formulado como fechamento máximo de um grafo (Lerchs e Grossmann, 1965): apesar de ter uma variável binária por bloco, ele pode ser resolvido de forma eficiente, sem Branch and Bound. Quando se acrescentam períodos e capacidades (o sequenciamento), essa propriedade se perde, e o problema volta a ser difícil.

> ***Resumo do capítulo:*** *o relaxamento linear dá um limite para o ótimo inteiro, mas a sua solução, arredondada, pode ser inviável ou ruim. Para achar e provar o ótimo inteiro, precisamos de um método organizado: o Branch and Bound.*

# Capítulo 4 — Branch and Bound para Programação Inteira

**Objetivos do capítulo:** entender as três etapas do Branch and Bound (ramificação, limitação e avaliação); aplicar os três testes de poda; resolver um problema de PI desenhando a árvore; conhecer as estratégias de busca e de ramificação usadas pelos solvers.

## 4.1 A ideia: dividir para conquistar

O Branch and Bound (em português, ramificação e avaliação progressiva, ou ramificação e limitação) não testa todas as combinações. Ele divide o problema em subproblemas menores, calcula para cada um o melhor resultado que ele *poderia* atingir e descarta os que não têm chance de superar a melhor solução já encontrada. Essa é a **enumeração implícita**: grupos inteiros de soluções são eliminados sem serem examinados um a um.

| Termo | Significado |
|---|---|
| Nó ou subproblema (SP) | O problema original mais as restrições acumuladas no caminho da árvore até ele |
| Limite | Z do relaxamento do nó: numa maximização, nenhuma solução inteira do nó passa dele |
| Titular (incumbente), Z* | A melhor solução inteira viável encontrada até o momento |
| Nó pendente | Nó que ainda pode conter uma solução melhor que a titular |
| Podar (avaliar, sondar) | Descartar um nó: ele não será mais dividido |
| Variável ramificada | A variável inteira fracionária escolhida para dividir o nó |

## 4.2 As três etapas

**Ramificação.** Se uma variável inteira xj tem valor fracionário xj* no relaxamento, criamos dois filhos:

$$x_j \leq \lfloor x_j^* \rfloor \qquad \text{ou} \qquad x_j \geq \lfloor x_j^* \rfloor + 1$$

Com x1* = 2,67, os ramos são x1 ≤ 2 e x1 ≥ 3. A faixa 2 < x1 < 3 é eliminada, mas ela não contém nenhum inteiro: nenhuma solução inteira se perde. Para uma binária fracionária, os ramos são simplesmente y = 0 e y = 1.

**Limitação.** Resolvemos o relaxamento linear de cada filho. O valor de Z é o limite do nó. Se o modelo for de PI pura com coeficientes inteiros no objetivo, o limite pode ser arredondado para baixo (Capítulo 3).

**Avaliação (poda).** Cada novo nó passa por três testes:

| Teste | Condição (maximização) | O que fazer |
|---|---|---|
| 1. Limite | Limite do nó ≤ Z* | Podar: o nó não pode superar a titular |
| 2. Inviabilidade | O relaxamento do nó não tem solução | Podar: sem solução real, sem solução inteira |
| 3. Solução inteira | A solução do relaxamento já é inteira | Podar; se Z > Z*, ela vira a nova titular |

Um nó que passa pelos três testes sem ser podado fica **pendente** e será dividido mais tarde. Quando não há mais nós pendentes, a titular é ótima. Se nunca houve titular, o problema não tem solução viável.

## 4.3 O algoritmo completo

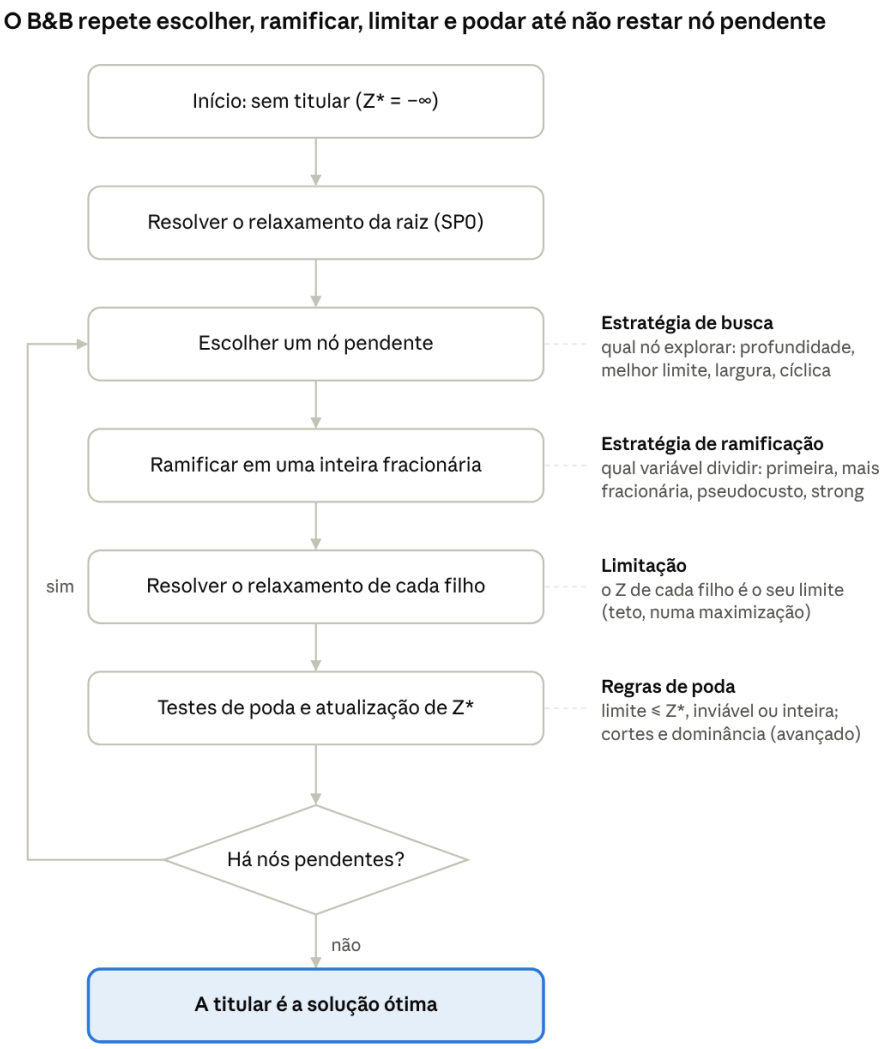



Figura 4.1 · Algoritmo de Branch and Bound, com os três componentes de Morrison et al. (2016)

Morrison, Jacobson, Sauppe e Sewell (2016) descrevem o Branch and Bound como um arcabouço com três componentes que o usuário pode configurar: a **estratégia de busca** (qual nó explorar), a **estratégia de ramificação** (como dividir o nó) e as **regras de poda** (como descartar nós). Os exemplos deste ebook usam a regra de Hillier e Lieberman:

1. **Escolha do nó:** entre os pendentes, o criado mais recentemente; em caso de empate, o de maior limite.
2. **Escolha da variável:** a primeira variável inteira fracionária, na ordem x1, x2, ...
1. **Avaliação:** os três testes, aplicados a cada filho assim que ele é criado.

## 4.4 Exemplo 4.1 — A frota do Exemplo 3.1, passo a passo

Voltamos ao problema da frota: max Z = 9x1 + 5x2, com orçamento 8x1 + 5x2 ≤ 53 e oficina 5x1 + 2x2 ≤ 26, x1 e x2 inteiras.

**Inicialização.** Z* = −∞. O relaxamento da raiz (SP0) dá (2,67; 6,33), com Z = 55,67. O limite é ⌊55,67⌋ = 55. A primeira variável fracionária é x1.

**Dica para resolver os nós.** Por real de orçamento, o caminhão grande rende 9/8 = 1,125 kt/dia e o médio, 5/5 = 1,0. Por hora de oficina, o grande rende 9/5 = 1,8 e o médio, 5/2 = 2,5. Em cada nó, o ótimo está num vértice: fixe a variável no limite imposto pelo ramo e use a outra até a restrição que aperta primeiro.

**Iteração 1 — ramificar SP0 em x1.**

| Nó | Restrições | Raciocínio | Solução | Z | Limite |
|---|---|---|---|---|---|
| SP1 | x1 ≤ 2 | x1 = 2; orçamento deixa x2 ≤ 7,4; oficina, x2 ≤ 8 | (2; 7,4) | 55 | 55 |
| SP2 | x1 ≥ 3 | x1 = 3; oficina deixa x2 ≤ 5,5 | (3; 5,5) | 54,5 | 54 |

Os dois são fracionários e não há titular: ambos ficam pendentes. Foram criados juntos, então o desempate é pelo maior limite: SP1.

**Iteração 2 — ramificar SP1 em x2 = 7,4.**

| Nó | Restrições | Solução | Z | Situação |
|---|---|---|---|---|
| SP3 | x1 ≤ 2; x2 ≤ 7 | (2; 7) | 53 | Inteira (Teste 3) → 1ª titular, Z* = 53 |
| SP4 | x1 ≤ 2; x2 ≥ 8 | (1,625; 8) | 54,625 | Limite 54 > 53 → pendente |

**Iteração 3 — ramificar SP4 (o mais recente) em x1 = 1,625.**

| Nó | Restrições | Solução | Z | Situação |
|---|---|---|---|---|
| SP5 | x1 ≤ 1; x2 ≥ 8 | (1; 9) | 54 | Inteira e 54 > 53 → nova titular, Z* = 54 |
| SP6 | x1 = 2; x2 ≥ 8 | — | — | Inviável: custo mínimo 8·2 + 5·8 = 56 > 53 (Teste 2) |

**Fechamento.** Resta SP2, com limite 54. Como 54 ≤ Z* = 54, ele é podado (Teste 1): no melhor caso empataria com a titular. Não há nós pendentes.

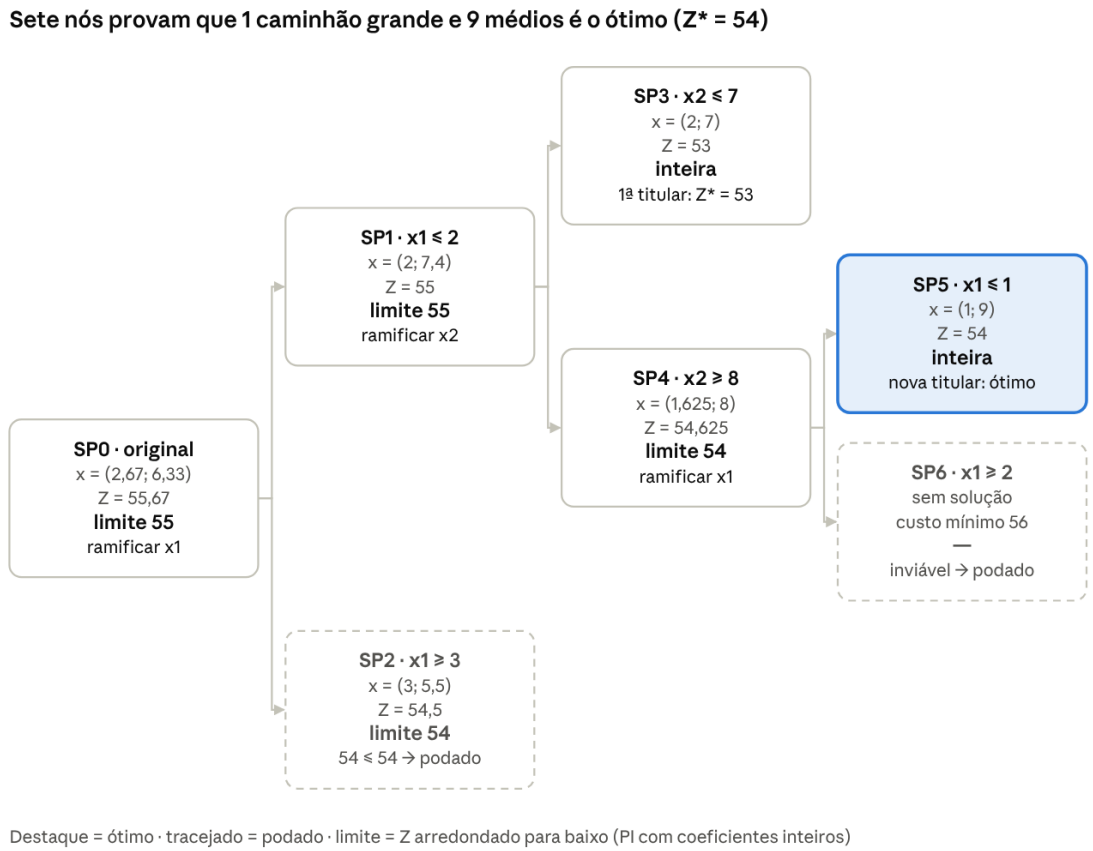



Figura 4.2 · Árvore do Exemplo 3.1, regra de Hillier e Lieberman

**Resposta:** comprar 1 caminhão grande e 9 médios, com capacidade de 54 kt/dia, custo de exatamente R$ 53 milhões e 23 h/dia de oficina. A árvore usou os três testes: inteira (SP3, SP5), inviável (SP6) e limite (SP2).

**Leitura geométrica.** Cada ramificação corta da região da Figura 3.1 uma faixa sem pontos inteiros ($2 < x1 < 3$, depois $7 < x2 < 8$, depois $1 < x1 < 2$). As sub-regiões ficam cada vez menores, até que o ótimo de cada uma caia sobre um ponto inteiro ou a região fique vazia.

## 4.5 Estratégias de busca: qual nó explorar?

A ordem de exploração não muda o ótimo encontrado, mas muda muito o número de nós e o tempo. Morrison et al. (2016) comparam quatro estratégias:

| Estratégia | Como escolhe | Vantagem | Desvantagem |
|---|---|---|---|
| Profundidade (DFS) | O nó mais recente (pilha) | Pouca memória; acha titulares cedo; reaproveita a solução do pai | Pode ficar presa num ramo ruim |
| Melhor limite (best-first) | O nó de maior limite | Explora o menor número de nós quando o limite é bom | Exige muita memória |
| Largura (BrFS) | Todos os nós de um nível antes do próximo | Simples | Demora a achar titulares; pouco usada sozinha |
| Cíclica (CBFS) | Agrupa os nós por profundidade e alterna, pegando o melhor de cada grupo | Combina as anteriores e diversifica a busca | Mais complexa de implementar |

Hillier e Lieberman explicam por que preferem o nó mais recente: o relaxamento do filho é quase igual ao do pai, então o solver reaproveita a solução anterior e reotimiza com poucas iterações do simplex dual. Esse reaproveitamento chama-se *warm start* (ou *hot start*, quando também se reaproveita a fatoração da base).

Morrison et al. também distinguem duas fases de todo Branch and Bound:

- **Fase de busca:** a solução ótima ainda não foi encontrada. A estratégia de busca é o que mais pesa aqui.
- **Fase de verificação:** a titular já é ótima, mas ainda há nós pendentes a podar para *provar* isso. Aqui, a força das regras de poda (limites apertados) é o que mais pesa.

No Exemplo 4.1, a fase de busca terminou em SP5; a fase de verificação foi só a poda de SP2. O algoritmo nunca sabe, durante a execução, em que fase está: só ao final se sabe que a titular encontrada em SP5 já era a ótima.

## 4.6 Estratégias de ramificação: qual variável dividir?

| Regra | Como escolhe a variável | Comentário |
|---|---|---|
| Primeira fracionária | A primeira na ordem do modelo | Didática; usada neste ebook |
| Mais fracionária | A de parte fracionária mais perto de 0,5 | Fácil, mas, segundo a literatura, não é melhor que uma escolha aleatória |
| Pseudocusto | Estima quanto Z cai ao ramificar, pelo histórico de ramificações anteriores | Rápida; fraca no início, quando não há histórico |
| Strong branching | Resolve de verdade os relaxamentos dos filhos para várias candidatas e escolhe a de maior efeito | Árvores menores, mas cada nó custa caro |
| Reliability branching | Usa strong branching até o pseudocusto de uma variável ficar confiável | O padrão de muitos solvers modernos |

Além da ramificação binária (dois filhos), existe a **ramificação larga**, que cria vários filhos de uma vez (por exemplo, um filho para cada modelo de britador). Ela reduz a profundidade da árvore, mas exige cuidado para não criar filhos demais nem filhos que se sobreponham.

A escolha da variável de ramificação é hoje tema de pesquisa com aprendizado de máquina. Strang e coautores (2025), por exemplo, usam aprendizado por reforço com planejamento (no estilo dos agentes que jogam xadrez e Go) para aprender regras de ramificação que geram árvores menores em famílias de problemas que se repetem. É uma leitura avançada, mas mostra que o tema continua vivo.

***Resumo do capítulo:*** *o Branch and Bound divide o problema em ramos sem perder soluções inteiras, usa o relaxamento de cada nó como limite e poda os nós que não podem melhorar a titular, que são inviáveis ou que já são inteiros. A ordem de exploração e a escolha da variável não mudam o ótimo, mas mudam o esforço.*

# Capítulo 5 — Branch and Bound para Programação Inteira Mista

**Objetivos do capítulo:** adaptar o Branch and Bound a modelos com variáveis contínuas; resolver à mão problemas de abertura de frentes e de dimensionamento de frota; acompanhar uma árvore em que a titular é trocada.

## 5.1 O que muda em relação à PI pura

O algoritmo é o mesmo, com quatro ajustes:

| Ponto | PI pura | PIM |
|---|---|---|
| Variáveis ramificadas | Qualquer variável fracionária | Somente as inteiras ou binárias fracionárias; as contínuas **nunca** são ramificadas |
| Ramos | $x_j \le \lfloor x_j^* \rfloor$ e $x_j \ge \lfloor x_j^* \rfloor + 1$ | Iguais; para binária, $y = 0$ e $y = 1$ |
| Limite | Pode ser arredondado para baixo (coeficientes inteiros) | Usado como está: Z pode ser fracionário |
| Teste 3 (solução inteira) | Todas as variáveis inteiras | Basta que as variáveis **declaradas** inteiras estejam inteiras; as contínuas podem ficar fracionárias |

## 5.2 Exemplo 5.1 — Abrir ou não uma nova frente

Uma pedreira de calcário lavra a frente A e estuda abrir a frente B, de calcário mais puro. Abrir B custa R$ 80 mil/mês (acessos, drenagem, mobilização), pagos só se B for lavrada. O britador recebe até 10 kt/mês.

| | Frente A | Frente B |
|---|---|---|
| Margem (R$ mil/kt) | 20 | 30 |
| Capacidade (kt/mês) | 8 | 6 |
| Custo fixo (R$ mil/mês) | — | 80 |

$$\max Z = 20x_1 + 30x_2 - 80y$$
$$\text{s.a.:}\quad x_1 \le 8;\quad x_2 \le 6y;\quad x_1 + x_2 \le 10$$
$$x_1, x_2 \ge 0;\quad y \in \{0,1\}$$

**Relaxamento (SP0).** Pela ligação x2 ≤ 6y, cada kt lavrado em B consome 1/6 de y, ou seja, paga 80/6 ≈ R$ 13,33 mil de custo fixo. A margem efetiva de B cai para 30 − 13,33 = 16,67, menor que os 20 de A. O relaxamento enche A (x1 = 8) e manda os 2 kt restantes para B (x2 = 2, y = 1/3). Z = 160 + 60 − 26,67 = 193,33. "Abrir um terço da frente" não existe: ramificamos em y.

| Nó | Decisão | Solução do relaxamento | Z | Situação |
|---|---|---|---|---|
| SP0 | — | x1 = 8; x2 = 2; y = 1/3 | 193,33 | Ramificar em y |
| SP1 | y = 0 | x1 = 8; x2 = 0 | 160 | y inteiro → titular |
| SP2 | y = 1 | x1 = 4; x2 = 6 | 180 | y inteiro e 180 > 160 → nova titular |

**Resposta:** abrir a frente B, lavrar 6 kt/mês em B e 4 kt/mês em A, com Z = R$ 180 mil/mês. A frente B, de maior margem, ocupa primeiro o britador, e A fica com 4 kt/mês de capacidade ociosa.

## 5.3 Exemplo 5.2 — Frota e produção

Cada kt entregue ao britador gera margem de R$ 15 mil. Cada caminhão alocado custa R$ 10 mil/dia e transporta até 1,5 kt/dia. O britador recebe até 11 kt/dia. x1 é o minério transportado (contínua) e x2 o número de caminhões (inteira).

$$\max Z = 15x_1 - 10x_2 \quad \text{s.a.:} \quad x_1 \leq 1{,}5x_2; \quad x_1 \leq 11; \quad x_1 \geq 0; \quad x_2 \in \{0, 1, 2, \dots\}$$

Como só x2 é inteira, a região viável da PIM não é um conjunto de pontos, mas de **segmentos horizontais**, um para cada número de caminhões.

| Nó | Restrição | Solução | Z | Situação |
|---|---|---|---|---|
| SP0 | — | x1 = 11; x2 = 7,33 | 91,67 | Limite 91,67 (sem arredondar); ramificar em x2 |
| SP1 | x2 ≤ 7 | x1 = 10,5; x2 = 7 | 87,5 | x2 inteira → titular (x1 = 10,5 pode ser fracionária) |
| SP2 | x2 ≥ 8 | x1 = 11; x2 = 8 | 85 | 85 ≤ 87,5 → podado |

**Resposta:** 7 caminhões levando 10,5 kt/dia, Z = R$ 87,5 mil/dia. O britador fica com 0,5 kt/dia ociosos, e isso é ótimo: um 8º caminhão acrescentaria só 0,5 kt/dia (+R$ 7,5 mil), mas custaria R$ 10 mil/dia.

| Decisão | Caminhões | Z (R$ mil/dia) |
|---|---|---|
| Relaxamento PL | 7,33 | 91,67 |
| Arredondar para cima | 8 | 85,0 |
| Branch and Bound | 7 | **87,5** |

## 5.4 Exemplo 5.3 — Quatro frentes com custo fixo (árvore de 11 nós)

A usina processa até 11 kt/semana. A frente A já está aberta; B, C e D só produzem se forem abertas.

| Frente | Margem (R$ mil/kt) | Capacidade (kt) | Custo fixo (R$ mil) | Margem efetiva (m − F/U) |
|---|---|---|---|---|
| A | 14 | 5 | — | 14 |
| B | 18 | 3 | 20 | 11,33 |
| C | 16 | 4 | 15 | 12,25 |
| D | 20 | 4 | 30 | 12,5 |

$$\max Z = 14x_A + 18x_B + 16x_C + 20x_D - 20y_B - 15y_C - 30y_D$$
$$\text{s.a.:} \quad x_A \leq 5; \quad x_B \leq 3y_B; \quad x_C \leq 4y_C; \quad x_D \leq 4y_D$$
$$x_A + x_B + x_C + x_D \leq 11; \quad x \geq 0; \quad y \in \{0{,}1\}$$

**Como resolver cada nó à mão.** No relaxamento, o modelo escolhe y = x/U, e cada kt de uma frente ainda livre paga F/U de custo fixo. Então:

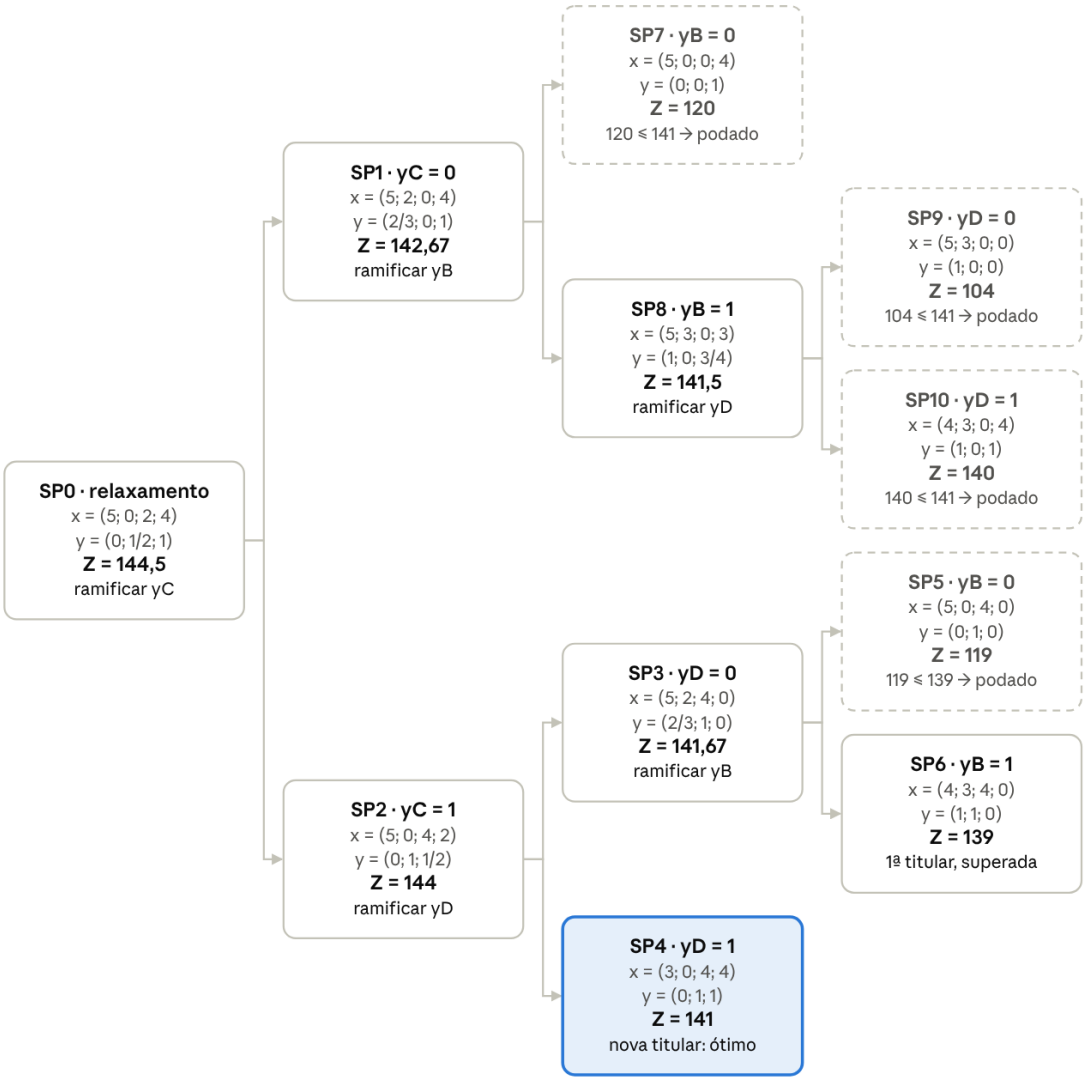
1. Frente com y fixado em 0: fica fora.
2. Frente com y fixado em 1: o custo fixo já está pago; ela entra com a margem cheia.
3. Frente com y livre: entra com a margem efetiva.
4. Encha a usina (11 kt) da maior margem para a menor, respeitando cada capacidade.

**Regra de navegação:** ramificar na primeira binária fracionária (ordem yB, yC, yD); explorar sempre os nós mais recentes e, entre irmãos, o de maior limite.

| Passo | Nó | Fixações | Produção no relaxamento | Z | O que acontece | Z* |
|---|---|---|---|---|---|---|
| 1 | SP0 | — | A 5; D 4; C 2 (yC = 1/2) | 144,5 | Ramificar em yC | — |
| 2 | SP2 | yC = 1 | C 4; A 5; D 2 (yD = 1/2) | 144 | Ramificar em yD | — |
| 3 | SP3 | yC = 1; yD = 0 | C 4; A 5; B 2 (yB = 2/3) | 141,67 | Ramificar em yB | — |
| 4 | SP6 | yC = 1; yD = 0; yB = 1 | B 3; C 4; A 4 | 139 | Inteira → 1ª titular | 139 |
| 5 | SP5 | yC = 1; yD = 0; yB = 0 | C 4; A 5 | 119 | 119 ≤ 139 → podado | 139 |
| 6 | SP4 | yC = 1; yD = 1 | D 4; C 4; A 3 | 141 | Inteira e 141 > 139 → nova titular | 141 |
| 7 | SP1 | yC = 0 | A 5; D 4; B 2 (yB = 2/3) | 142,67 | 142,67 > 141: ainda pode haver algo melhor → ramificar em yB | 141 |
| 8 | SP8 | yC = 0; yB = 1 | B 3; A 5; D 3 (yD = 3/4) | 141,5 | Ramificar em yD | 141 |

| Passo | Nó | Fixações | Produção no relaxamento | Z | O que acontece | Z* |
|---|---|---|---|---|---|---|
| 9 | SP10 | yC = 0; yB = 1; yD = 1 | D 4; B 3; A 4 | 140 | 140 ≤ 141 → podado | 141 |
| 10 | SP9 | yC = 0; yB = 1; yD = 0 | B 3; A 5 | 104 | 104 ≤ 141 → podado | 141 |
| 11 | SP7 | yC = 0; yB = 0 | A 5; D 4 | 120 | 120 ≤ 141 → podado | 141 |

**Onze nós até o ótimo: abrir C e D, Z* = R$ 141 mil/semana**




x = (xA; xB; xC; xD) em kt · y = (yB; yC; yD) · destaque = ótimo · tracejado = podado

Figura 5.1 · Árvore do Exemplo 5.3, busca em profundidade com o filho de maior limite primeiro

**Resposta:** abrir C e D (não abrir B) e lavrar A = 3, C = 4 e D = 4 kt/semana, com Z = R$ 141 mil/semana.

Três lições desta árvore:

- **A primeira solução inteira não é necessariamente a ótima.** SP6 (Z = 139) foi superada por SP4 (Z = 141). É a transição entre a fase de busca e a fase de verificação descrita no Capítulo 4.
- **Achar uma boa titular não encerra a busca.** SP1 ainda tinha limite 142,67 > 141, e todo o seu ramo precisou ser examinado para provar a otimalidade.
- **A intuição engana por pouco.** "Abrir as frentes de maior margem" (B e D) dá 140, "abrir tudo" dá 133. B tem margem alta, mas dilui mal o seu custo fixo: R$ 20 mil para só 3 kt.

***Resumo do capítulo:*** *na PIM, ramifica-se apenas nas inteiras e binárias; o limite não é arredondado; uma solução com toneladas fracionárias é aceita se as inteiras estiverem inteiras. A ideia de margem efetiva (m − F/U) explica como o relaxamento "divide" o custo fixo e por que ele superestima o lucro.*

# Capítulo 6 — Formulações fortes, cortes e o que os solvers fazem

**Objetivos do capítulo:** entender por que dois modelos corretos podem ter desempenhos muito diferentes; escolher o M; conhecer planos de corte, branch-and-cut e as outras técnicas usadas pelos solvers.

## 6.1 Formulação forte e formulação fraca

Um mesmo problema inteiro pode ser escrito de várias formas corretas: todas têm exatamente as mesmas soluções inteiras. Mas os seus relaxamentos são diferentes. Uma formulação é mais **forte** (ou mais apertada) quando a região do seu relaxamento é menor, ficando mais perto dos pontos inteiros. Na prática, isso significa:

- limites mais próximos do ótimo inteiro (gap de integralidade menor);
- mais nós podados pelo Teste 1;
- árvores menores e solução mais rápida.

A formulação ideal seria aquela cujo relaxamento é exatamente o menor polígono que contém todos os pontos inteiros (a **envoltória convexa**). Ela daria a solução inteira já no relaxamento, mas quase nunca é conhecida. Os cortes da seção 6.3 são uma forma de chegar mais perto dela.

## 6.2 A escolha do M

No Exemplo 5.1 (frente B com custo fixo de 80 e capacidade de 6 kt), compare três formas de escrever a ligação:

| Formulação da frente B | Relaxamento | Z do relaxamento | Solução inteira |
|---|---|---|---|
| x2 ≤ 6y (M = capacidade) | x1 = 8; x2 = 2; y = 0,33 | 193,3 | 180 (correta) |
| x2 ≤ 100y e x2 ≤ 6 | x1 = 4; x2 = 6; y = 0,06 | 255,2 | 180 (correta, mas limite fraco) |
| x2 ≤ 100y (sem x2 ≤ 6) | x1 = 0; x2 = 10; y = 0,1 | 292,0 | 220: **errada**, lavra 10 kt numa frente de 6 kt |

O motivo está na margem efetiva do Capítulo 5: no relaxamento, cada kt da frente paga F/M de custo fixo. Com M = 6, são 13,33 por kt; com M = 100, são só 0,80 por kt. Quanto maior o M, mais "barato" o relaxamento enxerga o custo fixo, e mais longe o limite fica do ótimo.

**Regras para o M:**

1. M deve ser o **menor valor que não elimina nenhuma solução válida**.
2. Use limites físicos: capacidade da frente, da usina, da frota, demanda de mercado.
3. Se o M faz também o papel de capacidade, não o troque sem repor a capacidade como restrição própria.
4. M exageradamente grande também causa problemas numéricos no solver (números muito diferentes na mesma restrição).

## 6.3 Planos de corte

Um **plano de corte** (ou desigualdade válida) é uma nova restrição que corta parte da região do relaxamento, inclusive a solução fracionária atual, **sem eliminar nenhuma solução inteira**.

### Exemplo 6.1 — Corte de cobertura na carteira de projetos

No Exemplo 2.1, o relaxamento dá x1 = x2 = 0,89 e x4 = 1, com Z = 16,44, enquanto o ótimo inteiro é 14. Observe a restrição de orçamento, 6x1 + 3x2 + 5x3 + 2x4 ≤ 10.

O grupo {britador, correia, pilha} custa 6 + 3 + 2 = 11 > 10: os três juntos não cabem. Mas, retirando qualquer um, o restante cabe (5, 8 ou 9). Esse grupo é uma **cobertura mínima** da restrição, e dele sai o corte "no máximo 2 dos 3":

$$x_1 + x_2 + x_4 \leq 2$$

Nenhuma carteira viável viola esse corte, mas a solução fracionária viola: 0,89 + 0,89 + 1 = 2,78. Com o corte, o relaxamento passa a dar **Z = 14**, exatamente o ótimo inteiro. O gap de integralidade caiu de 2,44 para zero com uma única restrição, e o Branch and Bound pode podar quase tudo de imediato.

Procedimento geral (Hillier e Lieberman, seção 11.8):

1. Tome uma restrição do tipo ≤ com coeficientes não negativos.
2. Ache um grupo de N variáveis que viola a restrição se todas valerem 1, mas a satisfaz se qualquer uma passar para 0.
3. O corte é: soma das variáveis do grupo ≤ N − 1.

Outros cortes famosos são os de **Gomory** (1958), obtidos do quadro final do simplex, e os cortes de arredondamento inteiro misto (MIR). Os primeiros algoritmos de PI usavam só cortes e não funcionavam bem na prática. A combinação de cortes com Branch and Bound é que se mostrou poderosa.

## 6.4 O que um solver moderno faz

Solvers como CBC, SCIP, CPLEX e Gurobi não executam o Branch and Bound "puro" deste ebook. Eles combinam:

| Técnica | O que faz | Exemplo |
|---|---|---|
| Pré-processamento | Antes de começar, fixa variáveis, elimina restrições redundantes e aperta coeficientes | Se a frente D não cabe em nenhuma solução viável, yD = 0 é fixada |
| Branch-and-cut | Acrescenta cortes na raiz e nos nós, apertando os relaxamentos | Cortes de cobertura, Gomory, MIR |
| Heurísticas primais | Procuram cedo boas soluções inteiras, para ter uma titular forte | Arredondamento inteligente, busca local |
| Regras de busca e de ramificação | Escolhem nós e variáveis com critérios sofisticados | Melhor limite combinado com profundidade; reliability branching |
| Relaxamento lagrangiano | Leva restrições difíceis para o objetivo com multiplicadores, gerando limites rápidos | Relaxar a capacidade por período no sequenciamento |
| Branch-and-price | Gera variáveis (colunas) sob demanda, por decomposição de Dantzig-Wolfe | Problemas com um número enorme de padrões possíveis |
| Dominância e simetria | Descartam nós equivalentes ou provadamente piores que outros | Caminhões idênticos: trocar o caminhão 3 pelo 5 não muda nada |
| Programação por restrições | Propaga regras lógicas para reduzir os domínios antes de calcular limites | Mariz, Peroni e Abreu Silva (2024), no agrupamento de blocos |

Morrison et al. (2016) resumem a diferença entre as duas grandes extensões: o **branch-and-cut** acrescenta restrições (linhas) para apertar a região; o **branch-and-price** acrescenta variáveis (colunas) que não cabiam no modelo inicial. Os dois podem ser combinados (branch-and-cut-and-price).

## 6.5 Lendo o relatório do solver

Ao resolver um modelo grande, o solver mostra a evolução de dois números: a melhor solução inteira (titular) e o melhor limite. O gap entre eles diminui com o tempo.

- **Gap = 0%:** otimalidade provada.
- **Limite de gap (por exemplo, 1%):** é comum parar quando o gap fica abaixo de uma tolerância. A solução está garantidamente a menos de 1% do ótimo.
- **Limite de tempo:** se o tempo acabar, o solver entrega a titular e o gap atual. Uma solução com gap de 3% pode ser muito útil na prática, mas não é ótima provada.

No planejamento de lavra, é normal trabalhar com gaps pequenos, mas diferentes de zero, porque os dados (teores, preços, recuperações) têm incerteza muito maior que 1%.

> ***Resumo do capítulo:*** *a qualidade da formulação importa tanto quanto o algoritmo. Use o menor M válido, prefira restrições que apertem o relaxamento e deixe o solver acrescentar cortes, heurísticas e pré-processamento. O gap mede a distância entre a melhor solução conhecida e o ótimo.*

# Capítulo 7 — PI e PIM no planejamento de lavra

**Objetivos do capítulo:** reconhecer onde a PI e a PIM aparecem nas etapas do planejamento de lavra; entender por que os problemas reais são grandes demais para uma formulação direta; conhecer as estratégias usadas na pesquisa para torná-los tratáveis.

## 7.1 Uma visão de conjunto

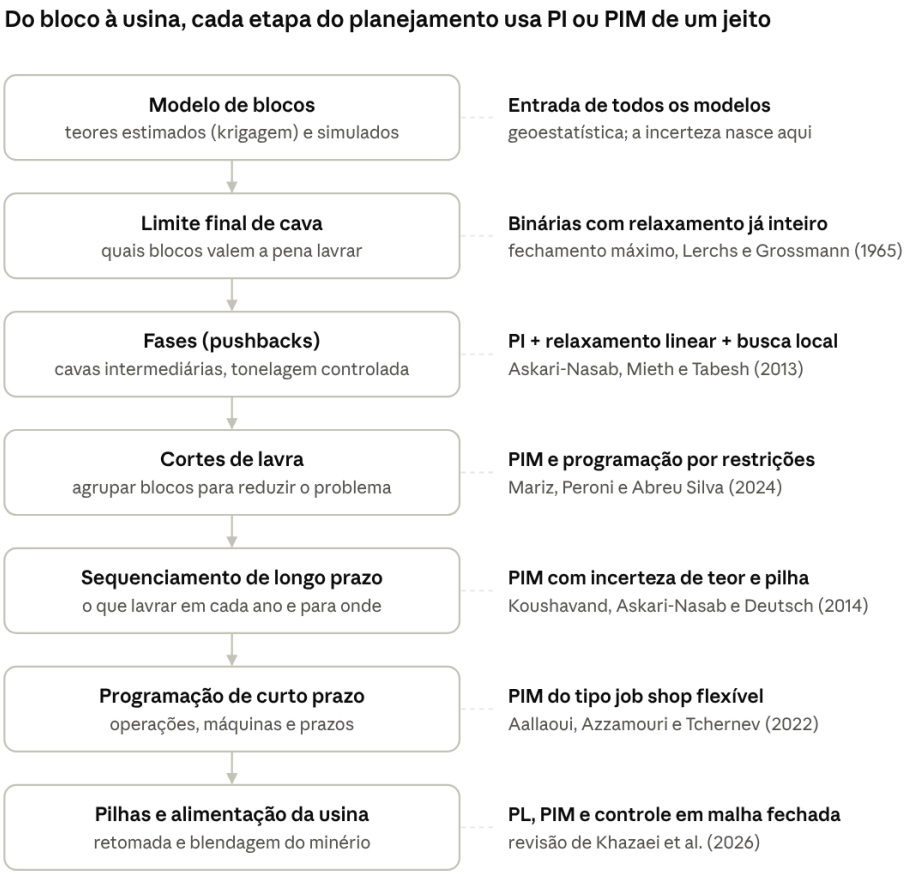



Figura 7.1 · Etapas do planejamento a céu aberto e as fontes deste capítulo

O planejamento de uma mina a céu aberto é dividido em etapas porque o problema completo ("o que lavrar, quando e para onde mandar, bloco a bloco, durante toda a vida da mina") é grande demais para ser resolvido de uma vez. Cada etapa usa os resultados da anterior e tem o seu próprio modelo.

## 7.2 Limite final de cava

O problema do limite final de cava escolhe o conjunto de blocos que maximiza o valor total, respeitando a precedência (para lavrar um bloco, é preciso lavrar os que estão acima dele). Cada bloco é uma variável binária, e a precedência é a restrição $x_b \leq x_a$ do Capítulo 2.

Lerchs e Grossmann (1965) mostraram que esse problema equivale ao **fechamento máximo de um grafo**, resolvido de forma exata e eficiente. Como vimos no Capítulo 3, é um caso em que o relaxamento linear já

tem solução inteira: não é preciso Branch and Bound. O método só ganhou implementação completa em 3D cerca de duas décadas depois (Whittle, 1988) e até hoje é a base das cavas aninhadas usadas para gerar fases.

## 7.3 Sequenciamento de longo prazo: o tamanho do problema

O sequenciamento acrescenta períodos e capacidades ao limite final de cava (modelo da seção 2.3). Com isso, a estrutura especial se perde, e o problema passa a ser **NP-difícil**: na forma de uma mochila com precedências, não há algoritmo conhecido que o resolva exatamente em tempo polinomial.

Mariz, Peroni e Abreu Silva (2024) lembram que depósitos de classe mundial têm milhões de blocos e, portanto, vários milhões de arcos de precedência, o que torna inviável a abordagem "monolítica" (um único modelo para tudo). As estratégias para reduzir o problema, usadas desde os anos 1960, são:

- calcular o limite final de cava e descartar os blocos fora dele;
- dividir a cava em fases (pushbacks);
- separar os horizontes de longo, médio e curto prazo;
- agregar blocos em unidades maiores: camadas, blocos-fase ou **cortes de lavra** (polígonos de blocos de uma mesma bancada).

## 7.4 Fases com PI e busca local (Askari-Nasab et al., 2013)

Askari-Nasab, Mieth e Tabesh (2013) propõem uma metodologia em três partes para minas a céu aberto:

1. **Desenho de fases controladas.** Uma formulação de PI define cada pushback com controle da tonelagem de minério e estéril. O **relaxamento linear** dessa formulação dá uma aproximação da solução e é usado para reduzir o tamanho do problema. Se o problema restante for pequeno, um solver de PI o resolve; se não, uma heurística gulosa cria uma solução inicial, que uma busca local melhora.
2. **Agrupamento.** Um agrupamento hierárquico, com controle de tamanho e forma, junta blocos em polígonos lavráveis dentro de cada fase.
3. **Sequenciamento.** Um modelo de PIM usa as fases e os agrupamentos como unidades de planejamento para gerar cronogramas práticos de vida da mina, quase ótimos, inclusive com várias cavas e vários processos, e resolve junto o teor de corte.

O estudo de caso é um depósito de minério de ferro. Repare como as ideias deste ebook aparecem: relaxamento linear como limite e como redutor do problema, PI exata quando o tamanho permite, heurísticas quando não permite.

## 7.5 Agrupamento de cortes de lavra com PIM e programação por restrições (Mariz et al., 2024)

No agrupamento, cada bloco de uma bancada é atribuído a um corte de lavra, buscando cortes parecidos (em teor e tipo de rocha) e com forma operacional. Mariz, Peroni e Abreu Silva (2024) formulam esse problema como uma **PIM** e o resolvem de três formas na bancada 15 do conjunto de dados Newman1:

| Abordagem | Resultado relatado |
|---|---|
| PIM exata (solver) | Não convergiu em tempo viável |
| Programação por restrições, satisfação (CSP) | 7 dias explorando o espaço; 74.581 soluções viáveis encontradas |
| Programação por restrições, otimização (COP) | Solução melhor que a melhor do CSP em 117,14 segundos |

O artigo mostra um ponto importante: um modelo de PIM pode estar correto e, mesmo assim, não ser resolvido em tempo útil. Nesses casos, outras técnicas de otimização (como a programação por restrições) podem trabalhar sobre a mesma formulação. Os autores também observam que a solução do CSP teve

melhor resultado econômico que a do COP, o que sugere recalibrar o índice de similaridade usado no objetivo.

## 7.6 Sequenciamento com incerteza de teor e pilha (Koushavand et al., 2014)

O teor de cada bloco é estimado (por krigagem), mas o teor real pode ser diferente. Um plano feito só com o modelo estimado pode, na prática, deixar a usina sem minério suficiente em alguns anos e com excesso em outros. Koushavand, Askari-Nasab e Deutsch (2014) formulam uma **PIM** que decide a sequência de lavra dos blocos de uma cava pré-definida e o destino de cada um, com dois objetivos:

$$\max \{\mathrm{VPL}_{\mathrm{estimado}} - \mathrm{DCoU}\}$$

- **VPL** calculado com o modelo de blocos estimado (krigado).
- **DCoU** (custo descontado da incerteza): soma, ao longo dos períodos, dos custos dos desvios em relação à meta de produção, avaliados nas várias simulações geoestatísticas do depósito.

O custo do desvio é **assimétrico**. Faltar minério (subprodução) significa usina ociosa e receita perdida. Sobrar minério (superprodução) significa estocar, remanusear e processar mais tarde, perdendo só parte do valor pelo desconto. Por isso a subprodução é penalizada mais fortemente.

Principais conclusões do artigo:

- A **pilha de estoque** reduz as faltas: o modelo passa a lavrar minério extra nos primeiros anos e a estocá-lo, diminuindo o risco de falta mais tarde.
- A **capacidade de lavra** ótima pode ser estimada considerando a incerteza: rodando o modelo para várias capacidades, calcula-se o custo da incerteza de cada uma.
- Para que o modelo seja tratável, os blocos são agrupados em **cortes de lavra**, o que reduz muito o número de variáveis binárias. O método é ilustrado num depósito de areias betuminosas no norte de Alberta, Canadá.

Atenção: o título do artigo fala em "linear programming model", mas o próprio texto apresenta o modelo como programação linear inteira mista.

## 7.7 Programação de curto prazo como job shop (Aallaoui et al., 2022)

No curto prazo, as decisões são operações elementares sobre os blocos, executadas por máquinas especializadas ou multifuncionais. Na mina de fosfato estudada, o estéril passa por preparação do terreno, perfuração, desmonte e decapeamento (com draglines e tratores), e o fosfato é empilhado e carregado nos caminhões. Aallaoui, Azzamouri e Tchernev (2022) modelam esse problema como uma **PIM** baseada no problema clássico de **job shop flexível com defasagens mínimas** entre operações. O modelo considera a sequência das operações, a compatibilidade e a disponibilidade das máquinas, os deslocamentos e as prioridades das camadas.

Os resultados foram obtidos com o solver CPLEX 12.7 em instâncias pequenas. A instância ilustrativa tem 3 parcelas a lavrar, cada uma com 7 operações (uma camada de estéril e uma de fosfato), e uma frota de 10 máquinas. Aqui as binárias representam "a operação i é feita pela máquina k" e "a operação i vem antes da j", enquanto os tempos de início são contínuos: um exemplo típico de PIM com restrições ou-ou (Ferramenta 3).

## 7.8 Pilhas e blendagem da alimentação (Khazaei et al., 2026)

Entre a mina e a usina ficam as pilhas e os pátios de ROM, que definem o teor e a variabilidade do material que chega ao britador. Khazaei, Noriega, Askari-Nasab e Pourrahimian (2026) revisam 27 fontes publicadas entre 1976 e 2025 e organizam o tema em cinco linhas de pesquisa:

1. teoria física da blendagem;

2. modelagem do estado das pilhas;
3. sequência de retomada e programação dos equipamentos;
4. otimização da blendagem da alimentação (onde a PL e a PIM aparecem);
5. reconciliação por sensores e controle em malha fechada.

A revisão conclui que faltam trabalhos que combinem, ao mesmo tempo, alta fidelidade espacial e otimização rigorosa, e que o tratamento da incerteza e o uso de sensores em tempo real ainda são pouco desenvolvidos. Entre as lacunas mais importantes estão a programação espacialmente explícita da retomada e formulações de blendagem multiatributo que continuem tratáveis em operações polimetálicas.

## 7.9 Do exemplo de sala de aula ao modelo real

| Aspecto | Exemplos deste ebook | Modelos reais |
|---|---|---|
| Variáveis inteiras | 1 a 3 | Milhares a milhões |
| Dados | Números redondos e certos | Teores estimados, preços, recuperações, incerteza geológica |
| Solução | À mão, com árvore desenhada | Solver comercial ou acadêmico, horas de processamento, gap de tolerância |
| Redução do problema | Desnecessária | Cava final, fases, horizontes, agregação em cortes |
| Método | Branch and Bound puro | Branch-and-cut, heurísticas, decomposição, programação por restrições |

A estrutura das decisões, porém, é a mesma: binárias para abrir, lavrar e escolher; inteiras para contar equipamentos; contínuas para toneladas e horas; e restrições de ligação, capacidade e precedência.

> ***Resumo do capítulo:*** *o planejamento de lavra é uma sequência de problemas de otimização, quase todos de PI ou PIM. O limite final de cava tem estrutura especial e é resolvido com eficiência; o sequenciamento é NP-difícil e exige redução do problema (fases, horizontes, agregação). A pesquisa atual trata da incerteza de teor, das pilhas e de métodos alternativos quando o solver não converge.*

# Capítulo 8 — Resolvendo no computador com Python e PuLP

**Objetivos do capítulo:** instalar o PuLP; escrever modelos de PI e PIM em Python; ler o resultado; obter o relaxamento linear; montar um pequeno modelo de sequenciamento com precedência.

## 8.1 Instalação

O PuLP é uma biblioteca gratuita de Python para modelagem de PL, PI e PIM. A série 3 (por exemplo, 3.3.2) já inclui o solver CBC, que executa Branch and Bound com cortes.

```
pip install "pulp==3.3.2"
```

> ***Atenção à versão.*** *O PuLP 4 mudou a forma de criar variáveis e não traz mais o CBC. Os códigos deste ebook e das aulas usam a sintaxe clássica (`pl.LpVariable(...)`, `pl.PULP_CBC_CMD`) e foram testados no PuLP 3.3.2. Quem usar a versão atual (PuLP 4.x) deve instalar o PuLP com o CBC, usando `pip install "pulp[cbc]"` (ou colocar um executável cbc no PATH), criar as variáveis com `m.add_variable("x1", lowBound=0, cat="Integer")` e trocar `pl.PULP_CBC_CMD(...)` por `pl.COIN_CMD(...)`, que aceita os mesmos argumentos (msg, timeLimit, gapRel); a classe PULP_CBC_CMD não existe mais. Outra opção é instalar o HiGHS (`pip install "pulp[highs]"`) e resolver com `m.solve(pl.HiGHS(msg=False))`.*

Os tipos de variável são definidos pelo argumento `cat`:

| Tipo | Como declarar |
|---|---|
| Contínua (padrão) | pl.LpVariable("x", lowBound=0) |
| Inteira | pl.LpVariable("n", lowBound=0, cat="Integer") |
| Binária | pl.LpVariable("y", cat="Binary") |

## 8.2 PI pura — a frota do Exemplo 3.1

```
import pulp as pl

m = pl.LpProblem("Frota", pl.LpMaximize)
x1 = pl.LpVariable("x1", lowBound=0, cat="Integer")  # caminhões grandes
x2 = pl.LpVariable("x2", lowBound=0, cat="Integer")  # caminhões médios

m += 9*x1 + 5*x2                                     # capacidade (kt/dia)
m += 8*x1 + 5*x2 <= 53, "orcamento"
m += 5*x1 + 2*x2 <= 26, "oficina"

m.solve(pl.PULP_CBC_CMD(msg=0))
print(pl.LpStatus[m.status])                         # Optimal
print(x1.value(), x2.value(), pl.value(m.objective))
# saída: 1.0 9.0 54.0
```

Repare na estrutura: a primeira expressão acrescentada com `+=` é a função objetivo; as seguintes, com `<=`, `>=` ou `==`, são restrições. O nome depois da vírgula é opcional, mas ajuda a ler relatórios.

## 8.3 PIM — a frente B do Exemplo 5.1

```
import pulp as pl

m = pl.LpProblem("Frente_B", pl.LpMaximize)
x1 = pl.LpVariable("x1", lowBound=0)  # produção da frente A (kt/mês)
```

```
x2 = pl.LpVariable("x2", lowBound=0)                    # produção da frente B (kt/mês)
y  = pl.LpVariable("y", cat="Binary")                   # 1 se a frente B for aberta

m           += 20*x1 + 30*x2 - 80*y
m              += x1 <= 8,              "cap_A"
m += x2 <= 6*y, "liga_B"                                # M = capacidade de B
m         += x1 + x2 <= 10,          "britador"

m.solve(pl.PULP_CBC_CMD(msg=0))
print(x1.value(), x2.value(), y.value(), pl.value(m.objective))
# saída: 4.0 6.0 1.0 180.0
```

## 8.4 Modelos com dados em dicionários — as quatro frentes do Exemplo 5.3

Em modelos maiores, os dados ficam em dicionários e as variáveis são criadas em grupo com `LpVariable.dicts`. O mesmo código serve para 4 ou 400 frentes.

```
import pulp as pl

margem = {"A": 14, "B": 18, "C": 16, "D": 20}           # R$ mil/kt
cap    = {"A": 5,  "B": 3,  "C": 4,  "D": 4}            # kt/semana
fixo   = {"B": 20, "C": 15, "D": 30}                    # R$ mil/semana
usina  = 11                                             # kt/semana

m = pl.LpProblem("Quatro_frentes", pl.LpMaximize)
x = pl.LpVariable.dicts("x", margem, lowBound=0)
y = pl.LpVariable.dicts("y", fixo, cat="Binary")

m += pl.lpSum(margem[f]*x[f] for f in margem) - pl.lpSum(fixo[f]*y[f] for f in fixo)
m                 += x["A"] <= cap["A"]
for f in fixo:
    m         += x[f] <= cap[f]*y[f],      f"liga_{f}"
m         += pl.lpSum(x.values()) <= usina,      "usina"

m.solve(pl.PULP_CBC_CMD(msg=0))
for f in margem:
    print(f, x[f].value(), y[f].value() if f in y else "-")
print("Z =", pl.value(m.objective))                     # Z = 141.0

# Relaxamento linear: o mesmo modelo com as binárias contínuas entre 0 e 1
for v in y.values():
    v.cat = pl.LpContinuous
m.solve(pl.PULP_CBC_CMD(msg=0))
print("Limite do relaxamento =", pl.value(m.objective))   # 144.5
```

A saída confirma a árvore do Capítulo 5: abrir C e D, com A = 3, C = 4 e D = 4 kt, Z = 141, e limite da raiz igual a 144,5. Trocar `cat` para contínua é a forma mais simples de ver o relaxamento e calcular o gap de integralidade.

## 8.5 Um pequeno sequenciamento com precedência

O modelo da seção 2.3, numa seção vertical com 6 blocos, 2 períodos e capacidade de 3 blocos por período. Os blocos de cima são estéril (valor negativo); o bloco do fundo é o mais rico.

```
import pulp as pl

# Seção vertical de uma cava: 3 bancadas, 6 blocos de mesma massa (1 unidade)
#                       bancada 1:  b1  b2  b3
#                       bancada 2:      b4  b5
#                       bancada 3:          b6
valor = {"b1": -2, "b2": -2, "b3": -2, "b4": 3, "b5": 4, "b6": 9}   # R$ mi (sem desconto)
acima = {"b4": ["b1", "b2"], "b5": ["b2", "b3"], "b6": ["b4", "b5"]}   # precedência
periodos = [1, 2]
capacidade = 3                                          # blocos por período
taxa = 0.10                                             # taxa de desconto por período
```

```
blocos                                    =                                      list(valor)
m                =                pl.LpProblem("Sequenciamento",                pl.LpMaximize)
x  =  pl.LpVariable.dicts("x",  [(b,  t)  for  b  in  blocos  for  t  in  periodos],  cat="Binary")

#    VPL:    valor    descontado    pelo    período    em    que    o    bloco    é    lavrado
m  +=  pl.lpSum(valor[b]  /  (1  +  taxa)**t  *  x[b,  t]  for  b  in  blocos  for  t  in  periodos)

for  b  in  blocos:                                          # cada bloco no máximo uma vez
    m      +=      pl.lpSum(x[b,      t]      for      t      in      periodos)      <=      1
for  t  in  periodos:                                          # capacidade de lavra
    m     +=     pl.lpSum(x[b,     t]     for     b     in     blocos)     <=     capacidade
for  b,  sup  in  acima.items():                                          #  precedência
    for                          a                          in                          sup:
        for                       t                       in                       periodos:
            m   +=   x[b,   t]   <=   pl.lpSum(x[a,   s]   for   s   in   periodos   if   s   <=   t)

m.solve(pl.PULP_CBC_CMD(msg=0))
for                           t                           in                           periodos:
    print("Período",    t,    [b    for    b    in    blocos    if    x[b,    t].value()    >    0.5])
print("VPL                =",                round(pl.value(m.objective),                3))
#                                                                                      saída:
#              Período              1              ['b2',              'b3',              'b5']
#              Período              2              ['b1',              'b4',              'b6']
# VPL = 8.264
```

Interpretação: o modelo lavra toda a seção, porque o valor total (−6 + 16 = 10) é positivo. No 1º período, abre caminho até b5 removendo b2 e b3 (resultado líquido zero); no 2º, remove b1 e chega a b4 e ao bloco rico b6. Repare que a precedência obriga a lavrar estéril antes de chegar ao minério, e que o desconto torna o mesmo bloco menos valioso no período 2. Experimente mudar a capacidade para 2 blocos por período e veja o que acontece.

## 8.6 Vendo o Branch and Bound por dentro: o notebook PI_PIM

O PuLP entrega a resposta, mas esconde a árvore. Para estudar o algoritmo, use o notebook **PI_PIM.ipynb**, que acompanha este ebook. Ele implementa o Branch and Bound do zero, em Python, usando o `scipy.optimize.linprog` (método HiGHS) apenas para resolver o relaxamento de cada nó. Toda a lógica de ramificar, limitar e podar está no código, visível e comentada.

| Parte do notebook | O que faz |
|---|---|
| `Problema` | Guarda o modelo: c, A_ub, b_ub, A_eq, b_eq, limites, inteiras (True/False por variável) e sentido ("max" ou "min") |
| `resolver_relaxacao` | Resolve o PL de um nó com os limites das variáveis daquele nó |
| `branch_and_bound` | Executa o algoritmo e imprime cada decisão (verbose=True) |
| `tabela_nos` | Monta a tabela com pai, ramo, solução, z e decisão de cada nó |
| `desenhar_arvore` | Desenha a árvore, colorindo os nós por destino (ramificado, ótimo, titular superada, podado por limite ou por inviabilidade) |
| `desenhar_regiao_2d` | Para 2 variáveis, mostra a região, os pontos inteiros, as soluções de cada nó e as faixas cortadas pela ramificação |
| `verificar_com_scipy` | Confere o resultado com o solver scipy.optimize.milp |

Uma variável binária é declarada como inteira com limites (0, 1), e a PIM é obtida marcando como `False` as variáveis contínuas. As quatro frentes do Exemplo 5.3 ficam assim:

```
frentes                                    =                                      Problema(
    c=[14,  18,  16,  20,  -20,  -15,  -30],                  #  xA,  xB,  xC,  xD,  yB,  yC,  yD
    A_ub=[[1,   0,   0,   0,     0,     0,     0],                                    #   xA   <=   5
          [0,   1,   0,   0,   -3,     0,     0],                                 #   xB   <=   3   yB
          [0,   0,   1,   0,     0,   -4,     0],                                 #   xC   <=   4   yC
          [0,   0,   0,   1,     0,     0,   -4],                                 #   xD   <=   4   yD
          [1,   1,   1,   1,     0,     0,     0]],                                      #   usina
```

```
    b_ub=[5,                0,                0,                0,                11],
    limites=[(0,             None)]*4             +            [(0,              1)]*3,
    inteiras=[False]*4  +  [True]*3,                          #  PIM:  toneladas  contínuas
    nomes=["xA",        "xB",        "xC",        "xD",        "yB",        "yC",        "yD"],
    sentido="max",
)
res    =    branch_and_bound(frentes,    selecao_no="profundidade",    selecao_var="primeira")
desenhar_arvore(res,                          "Quatro                          frentes")
verificar_com_scipy(frentes, res)          # z* = 141 nos dois
```

**Experimento: a regra muda o esforço, não o ótimo.** Rodando o notebook nos exemplos deste ebook com as três estratégias de busca e as duas regras de variável disponíveis:

| Exemplo | Regra de variável | Profundidade | Melhor limite | Largura | Ótimo |
|---|---|---|---|---|---|
| Frota (Exemplo 3.1) | Primeira fracionária | 9 nós | 9 nós | 11 nós | Z* = 54 |
| Frota (Exemplo 3.1) | Mais fracionária | 15 nós | 13 nós | 15 nós | Z* = 54 |
| Quatro frentes (Exemplo 5.3) | Primeira fracionária | 11 nós | 11 nós | 11 nós | Z* = 141 |

Na frota, a regra "mais fracionária" gerou mais nós que a simples "primeira", um bom exemplo da observação de Morrison et al. (2016) de que essa regra, em geral, não é melhor que escolher ao acaso.

**Por que a árvore do notebook pode ser diferente da do ebook?** As regras são parecidas, mas não idênticas:

- Na busca em profundidade, o notebook explora sempre primeiro o ramo "≤" (ou "= 0"). O Capítulo 5 explora primeiro o filho de maior limite.
- O notebook resolve o PL de um nó quando o retira da lista; Hillier e Lieberman avaliam os dois filhos assim que são criados.
- O notebook não arredonda o limite para baixo na PI pura (Capítulo 3), o que pode deixar alguns nós sem poda.

Nas quatro frentes, por explorar primeiro o ramo yC = 0, o notebook troca de titular três vezes (Z* = 120, depois 140, depois 141) antes de provar o ótimo. Compare essa árvore com a Figura 5.1: os nós são os mesmos, mas a ordem e as podas mudam. Esse é um excelente exercício para discutir a fase de busca e a fase de verificação.

## 8.7 Dicas práticas

| Situação | O que fazer |
|---|---|
| Ver o modelo escrito | `m.writeLP("modelo.lp")` gera um arquivo de texto com objetivo e restrições |
| Ver o log do Branch and Bound | `pl.PULP_CBC_CMD(msg=1)` mostra nós, cortes, titular e gap |
| Limitar o tempo | `pl.PULP_CBC_CMD(timeLimit=60)` (segundos) |
| Aceitar um gap | `pl.PULP_CBC_CMD(gapRel=0.01)` para parar com gap de 1% |
| Conferir o status | `pl.LpStatus[m.status]`: Optimal, Infeasible, Unbounded ou Not Solved |
| Binária com valor 0,9999 | Normal (tolerância numérica): use `round(v.value())` ou `v.value() > 0.5` |

> ***Resumo do capítulo:*** *com poucas linhas de Python, os modelos dos capítulos anteriores são resolvidos em frações de segundo. O solver executa o mesmo Branch and Bound das árvores desenhadas à mão, acrescido de cortes, heurísticas e pré-processamento. Compare sempre a saída do computador com o que você espera do problema.*

# Capítulo 9 — Exercícios propostos

**Como usar este capítulo:** resolva cada exercício antes de olhar a resposta. Os gabaritos completos, com árvores desenhadas, de oito problemas do mesmo tipo estão na lista **Exercícios Resolvidos — Programação Inteira Mista no Planejamento de Mina**. Para conferir as contas, use o PuLP (Capítulo 8) ou o notebook PI_PIM.

## 9.1 Enunciados

**P1 — Regras lógicas (Capítulo 2).** Sejam yA, yB, yC e yD binárias de abertura de quatro frentes, z2 a binária de instalação da correia 2 e xC a produção da frente C (kt/semana). Escreva:

1. Se a frente C for aberta, a correia 2 deve ser instalada.
2. No máximo duas frentes podem ser abertas.
3. As frentes B e D não podem ser abertas juntas.
4. A frente C, se aberta, produz entre 2 e 5 kt/semana.

**P2 — Branch and Bound em PI (Capítulo 4).** Uma mina escolhe quantas escavadeiras pequenas (x1) e carregadeiras (x2) alocar a uma frente. Cada escavadeira acrescenta 5 kt/dia de produção e cada carregadeira, 4 kt/dia. Escavadeira e carregadeira consomem 6 e 4 mil litros de diesel por dia (limite: 24 mil L/dia) e ocupam 1 e 2 vagas de manutenção preventiva (limite: 6 vagas). Maximize a produção com x1 e x2 inteiras. Resolva o relaxamento graficamente, teste os arredondamentos e aplique o Branch and Bound com a regra de Hillier e Lieberman.

**P3 — Frota e produção (Capítulo 5).** Cada kt entregue ao britador gera R$ 10 mil de margem. Cada caminhão custa R$ 6 mil/dia e transporta até 2,5 kt/dia. O britador recebe até 12 kt/dia. Quantos caminhões alocar?

**P4 — Frentes com custo fixo (Capítulo 5).** A usina recebe até 10 kt/semana. A frente A já está aberta (margem 10, capacidade 6). A frente B tem margem 15, capacidade 4 e custo fixo 25. A frente C tem margem 13, capacidade 5 e custo fixo 12 (margens e custos em R$ mil). Calcule as margens efetivas, resolva o relaxamento e encontre o ótimo.

**P5 — Cortes de cobertura (Capítulo 6).** Para a restrição de orçamento $5y1 + 4y2 + 3y3 + 2y4 \leq 9$, com binárias, encontre todas as coberturas mínimas de três variáveis e escreva os cortes correspondentes. O grupo {y1, y2} é uma cobertura?

**P6 — Escolha do M (Capítulo 6).** Uma frente tem capacidade de 8 kt/mês e o britador recebe 12 kt/mês. Um colega propôs três ligações: $x \leq 8y$; $x \leq 12y$; $x \leq 1000y$. Quais são válidas? Qual é a melhor? Alguma delas exige uma restrição adicional?

**P7 — Sequenciamento no computador (Capítulo 8).** No modelo da seção 8.5, reduza a capacidade para 2 blocos por período. Antes de rodar, preveja o que acontece. Depois, confira.

**P8 — Discussão (Capítulo 7).** Por que, no modelo de Koushavand et al. (2014), faltar minério na usina custa mais do que sobrar? Como a pilha de estoque altera a sequência de lavra nos primeiros anos?

## 9.2 Respostas

**P1.**

1. $yC \leq z2$
2. $yA + yB + yC + yD \leq 2$
3. $yB + yD \leq 1$

4. $2yC \leq xC \leq 5yC$

**P2.** Modelo: max Z = 5x1 + 4x2; 6x1 + 4x2 ≤ 24; x1 + 2x2 ≤ 6. O relaxamento dá (3; 1,5) com Z = 21. Arredondar para (3; 2) é inviável (diesel 26); (3; 1) dá 19.

| Nó | Restrições | Solução | Z | Situação |
|---|---|---|---|---|
| SP0 | — | (3; 1,5) | 21 | Limite 21; ramificar x2 (x1 já é inteira) |
| SP1 | x2 ≤ 1 | (3,33; 1) | 20,67 | Limite 20 |
| SP2 | x2 ≥ 2 | (2; 2) | 18 | Inteira → titular 18 |
| SP3 | x2 ≤ 1; x1 ≤ 3 | (3; 1) | 19 | Inteira → titular 19 |
| SP4 | x2 ≤ 1; x1 ≥ 4 | (4; 0) | 20 | Inteira → titular 20 = limite de SP1: ótimo |

Ótimo: 4 escavadeiras e nenhuma carregadeira, 20 kt/dia. O ótimo não é vizinho do ponto do relaxamento.

**P3.** O relaxamento dá x = 12 kt e n = 4,8 caminhões, Z = 91,2. Ramos: n ≤ 4 dá x = 10 e Z = 76; n ≥ 5 dá x = 12 e Z = 90. Ótimo: 5 caminhões, Z = R$ 90 mil/dia. O 5º caminhão acrescenta 2 kt (R$ 20 mil) e custa R$ 6 mil.

**P4.** Margens efetivas: A = 10; B = 15 − 25/4 = 8,75; C = 13 − 12/5 = 10,6. O relaxamento enche C (5 kt) e depois A (5 kt): yC = 5/5 = 1 e yB = 0, já inteiras. **O relaxamento já é a solução ótima**: abrir só C, com Z = 65 − 12 + 50 = R$ 103 mil/semana. Confira: abrir só B dá 95; abrir B e C, 98; não abrir nenhuma, 60.

**P5.** As coberturas mínimas de três variáveis são {y1, y2, y3} (soma 12), {y1, y2, y4} (soma 11) e {y1, y3, y4} (soma 10). Os cortes são y1 + y2 + y3 ≤ 2, y1 + y2 + y4 ≤ 2 e y1 + y3 + y4 ≤ 2. O grupo {y1, y2} não é cobertura: 5 + 4 = 9 não excede 9. O grupo {y2, y3, y4} também não: soma 9.

**P6.** As três ligações são válidas para ligar x a y, mas só x ≤ 8y impõe a capacidade da frente. Com M = 12 ou M = 1000, é preciso acrescentar x ≤ 8. A melhor é x ≤ 8y: menor M, limite mais forte, uma restrição a menos.

**P7.** O modelo não lavra nenhum bloco (VPL = 0). Com 2 blocos por período, cabem só 4 blocos em 2 períodos, e chegar a b6 exige lavrar os 6. As sequências possíveis com 4 blocos têm VPL negativo. Por exemplo, b2 e b3 no período 1 e b5 no período 2 dão −4/1,1 + 4/1,21 ≈ −0,33. A capacidade de lavra decide se o minério do fundo pode ser alcançado.

**P8.** Faltar minério deixa a usina ociosa: a receita daquele período se perde. Sobrar minério gera custos de estocagem e remanuseio e adia a receita, perdendo só parte do valor pelo desconto. Por isso o custo de subprodução é maior (penalidade assimétrica). Com a pilha, o modelo prefere lavrar minério extra nos primeiros anos e estocá-lo, reduzindo o risco de faltas nos anos seguintes.

# Questionário de revisão

## Perguntas de resposta curta

1. Na frota do Exemplo 3.1, por que a solução arredondada (3; 6) não é viável?
2. Numa maximização, por que o ótimo do relaxamento linear é um limite superior para o ótimo inteiro?
3. Quando é permitido arredondar o limite de um nó para baixo?
4. Como escrever "o projeto 2 só pode ser aprovado se o projeto 1 for aprovado"?
5. Numa PIM, uma tonelagem contínua pode ficar fracionária quando todas as variáveis inteiras já estão inteiras?
6. Por que um M muito grande prejudica a formulação?
7. Por que uma solução inteira encontrada não prova, sozinha, que é ótima?
8. No Exemplo 5.2, por que 7 caminhões são melhores que 8?
9. Quais são os três componentes configuráveis do Branch and Bound, segundo Morrison et al. (2016)?
10. O que distingue a fase de busca da fase de verificação? Que componente pesa mais em cada uma?
11. Qual é a diferença entre pseudocusto e strong branching? O que é reliability branching?
12. Qual é a diferença entre branch-and-cut e branch-and-price?
13. O que é o DCoU no modelo de Koushavand et al. (2014), e por que a penalidade é assimétrica?
14. Para que serve agrupar blocos em cortes de lavra antes de resolver o sequenciamento?
15. Por que o limite final de cava pode ser resolvido sem Branch and Bound, mas o sequenciamento não?

## Gabarito

1. Custa 8·3 + 5·6 = R$ 54 milhões, acima do orçamento de R$ 53 milhões.
2. O relaxamento contém todas as soluções inteiras e ainda outras; o máximo sobre um conjunto maior não pode ser menor que o máximo sobre o subconjunto.
3. Quando Z de toda solução inteira for inteiro: todas as variáveis inteiras e todos os coeficientes do objetivo inteiros (PI pura). Na PIM, em geral não, porque as variáveis contínuas podem tornar Z fracionário; a regra só vale se Z assumir valores inteiros em todas as soluções viáveis (por exemplo, quando as variáveis contínuas não entram na função objetivo).
4. $x2 \leq x1$, com x1 e x2 binárias.
5. Sim. Só as variáveis declaradas inteiras precisam ser inteiras (Teste 3 da PIM).
6. No relaxamento, o custo fixo é "dividido" por M: com M grande, a abertura parece quase gratuita, o limite sobe, as podas pioram e a árvore cresce. Além disso, pode causar problemas numéricos; e, se M fazia o papel de capacidade, trocá-lo muda o problema.
7. Pode haver nó pendente com limite maior que a titular. Só quando nenhum nó pendente puder superá-la a otimalidade está provada (Exemplo 5.3: 139 parecia boa, mas o ótimo era 141).
8. O 8º caminhão custa R$ 10 mil/dia e só acrescenta 0,5 kt/dia (até o limite do britador), ou seja, R$ 7,5 mil/dia de margem.
9. Estratégia de busca (qual nó explorar), estratégia de ramificação (como dividir o nó) e regras de poda (como descartar nós).
10. Na fase de busca, o ótimo ainda não foi encontrado; na de verificação, ele já é a titular, mas ainda há nós a podar para prová-lo. A estratégia de busca pesa mais na primeira; as regras de poda (limites fortes), na segunda. A estratégia de ramificação afeta as duas.
11. O pseudocusto estima o efeito de ramificar numa variável pelo histórico de ramificações anteriores; o strong branching resolve de fato os relaxamentos dos filhos das candidatas. O reliability branching usa strong branching até que o pseudocusto de uma variável tenha histórico suficiente para ser confiável.

12. O branch-and-cut acrescenta restrições (cortes) para apertar o relaxamento; o branch-and-price acrescenta variáveis (colunas) geradas sob demanda, por decomposição de Dantzig-Wolfe.
13. O custo descontado da incerteza: a soma, ao longo dos períodos e das simulações geoestatísticas, dos custos de desvio em relação à meta de produção. É assimétrica porque faltar minério (usina ociosa, receita perdida) custa mais que sobrar (estocar, remanusear e processar depois).
14. Para reduzir drasticamente o número de variáveis binárias e de restrições de precedência, tornando o modelo de PIM tratável pelos solvers.
15. O limite final de cava é um problema de fechamento máximo, cujo relaxamento já tem solução inteira. O sequenciamento acrescenta períodos e capacidades, perde essa estrutura e se torna NP-difícil.

## Questões dissertativas

As questões abaixo pedem análise e não têm gabarito único.

1. **Componentes do Branch and Bound.** Discuta como a estratégia de busca, a estratégia de ramificação e as regras de poda dependem umas das outras. Dê um exemplo em que uma regra de poda forte torna menos importante a escolha da estratégia de busca.
2. **Formulação e desempenho.** Usando os Exemplos 5.1 e 6.1, explique por que dois modelos corretos do mesmo problema podem exigir árvores de tamanhos muito diferentes.
3. **Risco geológico.** Explique como o uso de simulações geoestatísticas, junto com o modelo krigado e uma pilha de estoque, pode mudar a sequência de lavra de longo prazo, buscando equilíbrio entre o VPL e o custo da incerteza.
4. **Quando o solver não converge.** Com base em Mariz, Peroni e Abreu Silva (2024) e em Askari-Nasab, Mieth e Tabesh (2013), discuta as alternativas quando um modelo de PIM correto não é resolvido em tempo útil: redução do problema, heurísticas, busca local, programação por restrições.
5. **Aprendizado de máquina no Branch and Bound.** A partir de Strang et al. (2025), discuta por que aprender regras de ramificação pode ser vantajoso em operações que resolvem repetidamente problemas parecidos, como o planejamento de curto prazo de uma mina.

# Glossário

| Termo | Definição | Onde aparece |
|---|---|---|
| Branch and Bound (B&B) | Método exato que divide o problema em subproblemas organizados numa árvore, calcula limites e poda os que não podem melhorar a titular | Cap. 4 e 5 |
| Branch-and-cut | B&B que acrescenta planos de corte aos relaxamentos dos nós | Cap. 6 |
| Branch-and-price | B&B que gera variáveis (colunas) sob demanda, por decomposição de Dantzig-Wolfe | Cap. 6 |
| Busca em profundidade (DFS) | Explora o nó criado mais recentemente; usa pouca memória e reaproveita a solução do pai | Cap. 4 |
| Busca pelo melhor limite (best-first) | Explora o nó pendente com o melhor limite; costuma examinar menos nós, mas usa mais memória | Cap. 4 |
| Busca cíclica (CBFS) | Agrupa os nós por profundidade e alterna entre os grupos, pegando o melhor de cada um | Cap. 4 |
| Cobertura mínima | Grupo de binárias que viola uma restrição se todas valerem 1, mas a satisfaz se qualquer uma valer 0; gera um corte | Cap. 6 |
| Corte (plano de corte, desigualdade válida) | Restrição que elimina soluções fracionárias do relaxamento sem eliminar soluções inteiras | Cap. 6 |
| Corte de lavra (mining cut) | Agrupamento de blocos vizinhos e parecidos de uma bancada, usado como unidade de planejamento para reduzir o modelo | Cap. 7 |
| Custo descontado da incerteza (DCoU) | Custo, trazido a valor presente, dos desvios em relação à meta de produção, calculado sobre simulações do depósito | Cap. 7 |
| Custo fixo | Custo pago só se uma atividade acontecer (abrir frente, mobilizar equipe), modelado com $F \cdot y$ e $x \leq M \cdot y$ | Cap. 2 |
| Envoltória convexa | Menor região convexa que contém todas as soluções inteiras; a formulação ideal | Cap. 6 |
| Enumeração implícita | Descartar grupos inteiros de soluções sem examiná-las uma a uma | Cap. 1 e 4 |
| Fase de busca / fase de verificação | Antes / depois de o ótimo se tornar a titular; na verificação, só resta provar a otimalidade | Cap. 4 |
| Fechamento máximo | Formulação do limite final de cava em grafos (Lerchs e Grossmann), resolvida com eficiência | Cap. 3 e 7 |
| Formulação forte | Formulação cujo relaxamento fica mais perto das soluções inteiras, dando limites melhores | Cap. 6 |
| Gap de integralidade | Diferença entre o ótimo do relaxamento e o ótimo inteiro | Cap. 3 |
| Gap de otimalidade | Diferença relativa entre o melhor limite e a titular durante a busca; 0% = ótimo provado | Cap. 3 e 6 |
| Limite (bound) | Melhor valor que um nó poderia atingir; numa maximização, o Z do seu relaxamento | Cap. 3 e 4 |
| M grande (Big-M) | Constante usada para ligar ou desligar restrições com binárias; deve ser o menor valor válido | Cap. 2 e 6 |
| Margem efetiva | Margem menos o custo fixo por unidade de capacidade ($m - F/U$); explica o relaxamento de modelos com custo fixo | Cap. 5 |
| Nó pendente | Subproblema ainda não podado que pode conter uma solução melhor que a titular | Cap. 4 |
| NP-difícil | Classe de problemas para os quais não se conhece algoritmo exato de tempo polinomial | Cap. 1 e 7 |
| PI, PIB, PIM (MIP, MILP) | Programação inteira pura, binária e inteira mista | Cap. 1 |
| Poda (avaliação, sondagem) | Descartar um nó por limite, por inviabilidade ou por solução inteira | Cap. 4 |
| Pré-processamento | Reformulações automáticas antes da busca: fixar variáveis, | Cap. 6 |

| Termo | Definição | Onde aparece |
|---|---|---|
| | eliminar redundâncias, apertar coeficientes | |
| Programação por restrições (CP) | Abordagem que propaga regras lógicas para reduzir domínios; usada por Mariz et al. (2024) | Cap. 6 e 7 |
| Pseudocusto | Estimativa, pelo histórico da árvore, de quanto Z muda ao ramificar numa variável | Cap. 4 |
| Ramificação larga | Ramificação que cria mais de dois filhos de uma vez | Cap. 4 |
| Relaxamento lagrangiano | Leva restrições difíceis para o objetivo com multiplicadores, gerando limites rápidos | Cap. 6 |
| Relaxamento linear | O modelo inteiro sem as condições de integralidade | Cap. 3 |
| Reliability branching | Usa strong branching até o pseudocusto de cada variável ficar confiável | Cap. 4 |
| Strong branching | Testa resolvendo os filhos de várias candidatas e escolhe a variável de maior efeito | Cap. 4 |
| Subprodução / superprodução | Produção abaixo / acima da meta da usina; a primeira custa mais | Cap. 7 |
| Teste de mesa | Substituir 0 e 1 nas binárias e verificar se cada restrição faz sentido | Cap. 2 |
| Titular (incumbente), Z* | Melhor solução inteira viável encontrada até o momento | Cap. 4 |
| Warm start / hot start | Reaproveitar a solução (e a fatoração) do relaxamento do pai para resolver o filho mais rápido | Cap. 4 |

# Referências

## Livro-texto

HILLIER, Frederick S.; LIEBERMAN, Gerald J. *Introdução à pesquisa operacional*. 8. ed. São Paulo: McGraw-Hill, 2006. Capítulo 11 (Programação Inteira), em especial as seções 11.6 (ramificação e avaliação progressiva para PIB), 11.7 (algoritmo para PIM) e 11.8 (ramificação e corte).

## Branch and Bound e métodos de solução

MORRISON, David R.; JACOBSON, Sheldon H.; SAUPPE, Jason J.; SEWELL, Edward C. Branch-and-bound algorithms: a survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, v. 19, p. 79–102, 2016. DOI: 10.1016/j.disopt.2016.01.005.

STRANG, Paul; ALÈS, Zacharie; BISSUEL, Côme; JUAN, Olivier; KEDAD-SIDHOUM, Safia; RACHELSON, Emmanuel. Planning in Branch-and-Bound: model-based reinforcement learning for exact combinatorial optimization. *arXiv*:2511.09219v1, 2025. Preprint; leitura avançada.

## Planejamento de lavra

AALLAOUI, S.; AZZAMOURI, A.; TCHERNEV, N. Mixed integer linear programming model for open pit mine scheduling. *IFAC-PapersOnLine*, v. 55, n. 10, p. 2276–2281, 2022. DOI: 10.1016/j.ifacol.2022.10.047.

ASKARI-NASAB, Hooman; MIETH, Clemens; TABESH, Mohammad. A hybrid local search/mixed integer programming approach to open pit controlled phase-design. In: APCOM, 2013.

KHAZAEI, Soroush; NORIEGA, Roberto; ASKARI-NASAB, Hooman; POURRAHIMIAN, Yashar. Stockpile reclamation and grade blending for processing plant feed: a systematic review of methods, models, and research gaps. *Mining*, v. 6, art. 62, 2026. DOI: 10.3390/mining6030062.

KOUSHAVAND, Behrang; ASKARI-NASAB, Hooman; DEUTSCH, Clayton V. A linear programming model for long-term mine planning in the presence of grade uncertainty and a stockpile. *International Journal of Mining Science and Technology*, v. 24, p. 451–459, 2014. DOI: 10.1016/j.ijmst.2014.05.006.

MARIZ, Jorge Luiz Valença; PERONI, Rodrigo de Lemos; ABREU SILVA, Ricardo Martins de. A constraint programming approach to solve the clustering problem in open-pit mine planning. *REM – International Engineering Journal*, v. 77, n. 2, e230060, 2024. DOI: 10.1590/0370-44672023770060.

LERCHS, H.; GROSSMANN, I. F. Optimum design of open-pit mines. *Transactions of the Canadian Institute of Mining*, v. 58, p. 17–24, 1965. Citado a partir de Mariz et al. (2024).

## Materiais da disciplina

MAIOR, Gleicon. *Pesquisa Operacional*: aula 4.1 (Programação Inteira) e aula 4.2 (Programação Inteira Mista). IFES, Campus Cachoeiro de Itapemirim. Apresentações de aula.

MAIOR, Gleicon. *Exercícios Resolvidos — Programação Inteira Mista no Planejamento de Mina* e *Exercícios de Branch and Bound — Planejamento de Mina*. Listas de exercícios da disciplina.

MAIOR, Gleicon. *PI_PIM.ipynb*: implementação didática do Branch and Bound em Python. Notebook da disciplina.

MAIOR, Gleicon. *Tutorial: Programação Inteira, Programação Inteira Mista e Branch and Bound*. Material de estudo que deu origem a este ebook.

## Software

PuLP: biblioteca Python de modelagem para PL, PI e PIM, com o solver CBC (série 3). SciPy: `scipy.optimize.linprog` e `scipy.optimize.milp` (solver HiGHS).

---

*Todos os exemplos numéricos deste ebook foram criados para fins didáticos e conferidos em Python (PuLP 3.3.2 e SciPy). Os artigos foram resumidos em paráfrase; consulte os originais para formulações, hipóteses e resultados completos.*