# Programação linear inteira

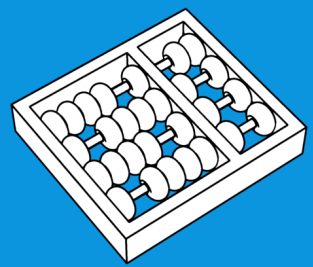
MC558 - Projeto e Análise de Algoritmos II

11/24

22

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

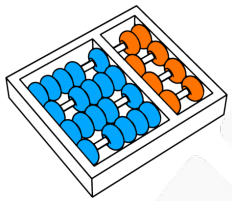
UNICAMP



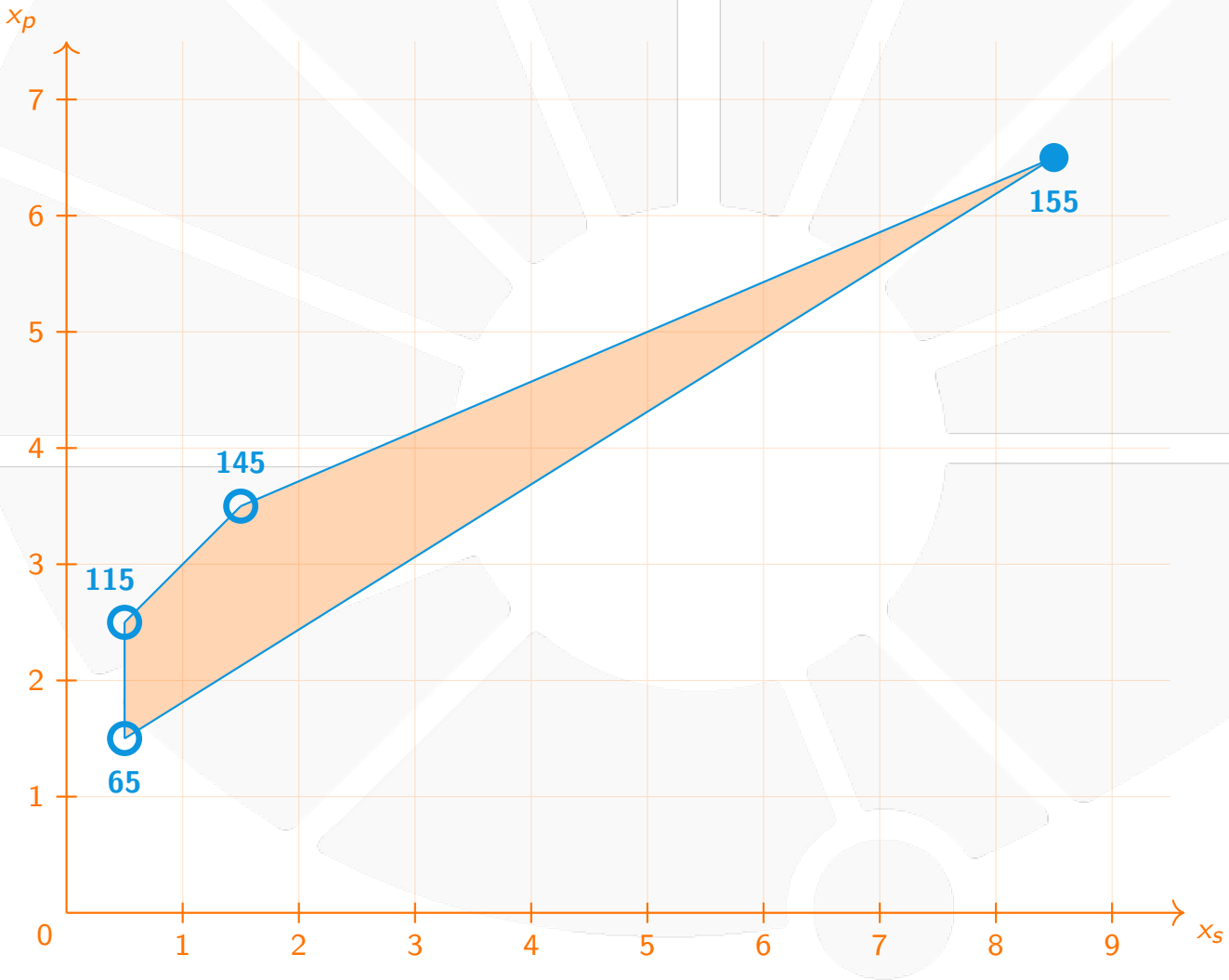
*"A arte de formular programas lineares e lineares inteiros é, bem, uma arte: é difícil de ensinar e ainda mais difícil de aprender."*

Gerald G. Brown e Robert F. Dell.

# Aulas anteriores



# Limpando o planeta




$$x_p, x_s \geq 0$$

$$x_s \geq 0.5$$

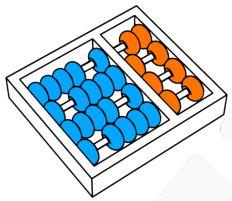
$$x_p - x_s \leq 2$$

$$7x_p - 3x_s \leq 20$$

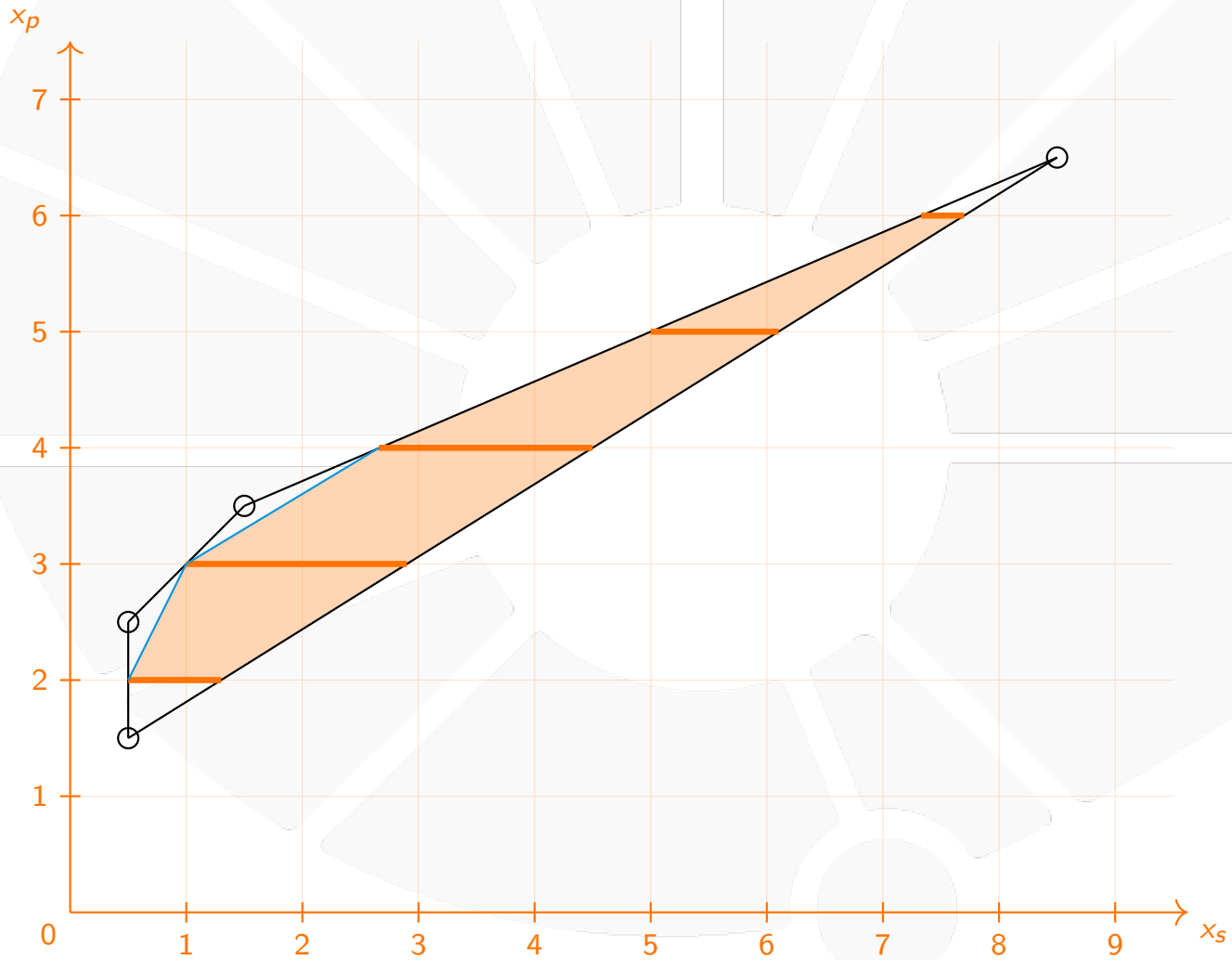
$$16x_p - 10x_s \geq 19$$

Função objetivo:

$$\max \quad 50x_p - 20x_s$$



# Limpando o planeta




$\mathbf{x_p} \in \mathbb{Z}, \mathbf{x_s} \in \mathbb{Q}$

$\mathbf{x_p}, \mathbf{x_s} \geq 0$

$\mathbf{x_s} \geq 0.5$

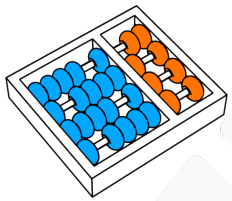
$\mathbf{x_p} - \mathbf{x_s} \leq 2$

$7\mathbf{x_p} - 3\mathbf{x_s} \leq 20$

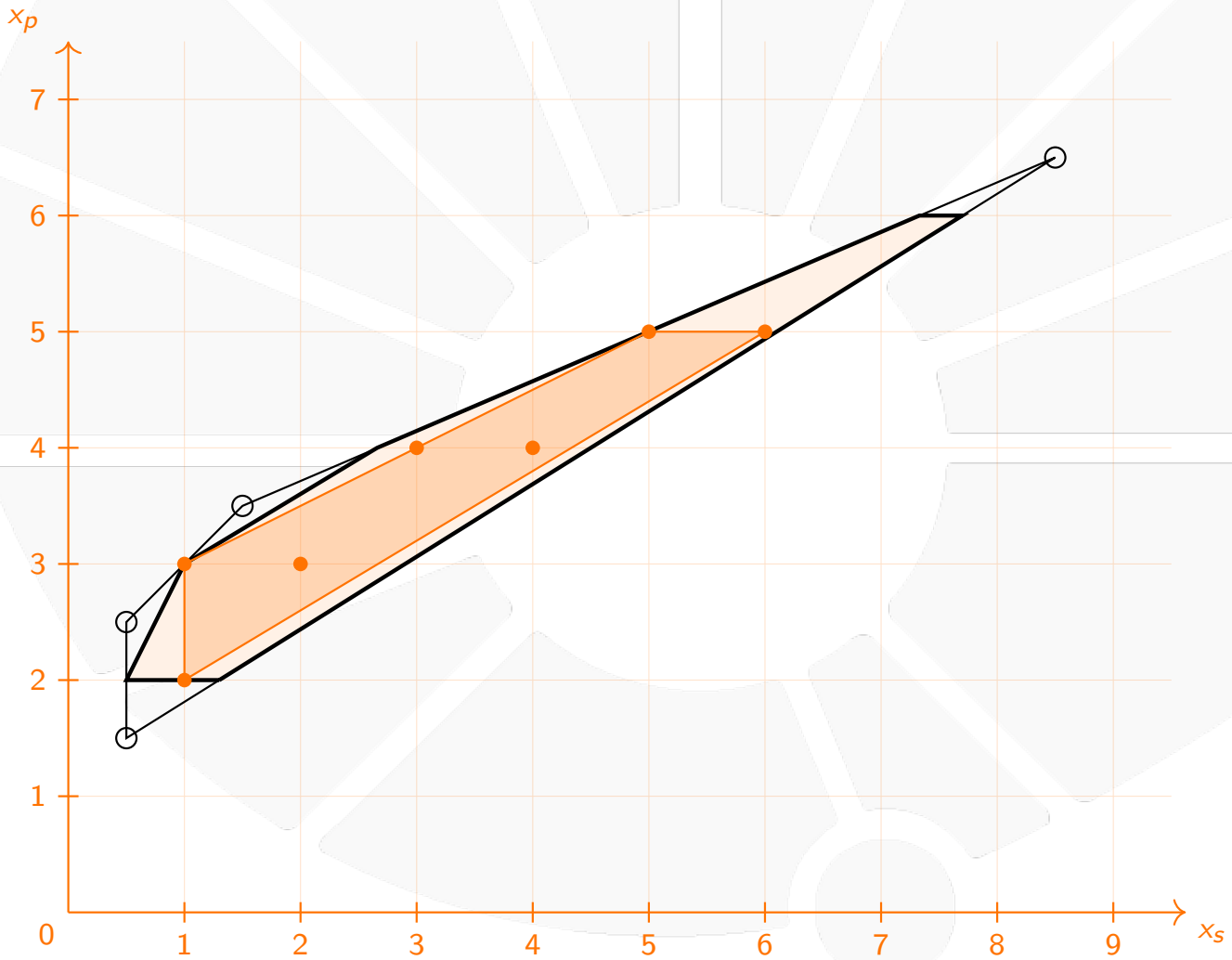
$16\mathbf{x_p} - 10\mathbf{x_s} \geq 19$

Função objetivo:

$\max \quad 50\mathbf{x_p} - 20\mathbf{x_s}$



# Limpando o planeta




$$x_p, x_s \in \mathbb{Z}$$

$$x_p, x_s \geq 0$$

$$x_s \geq 0.5$$

$$x_p - x_s \leq 2$$

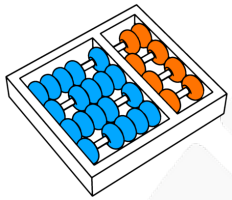
$$7x_p - 3x_s \leq 20$$

$$16x_p - 10x_s \geq 19$$

Função objetivo:

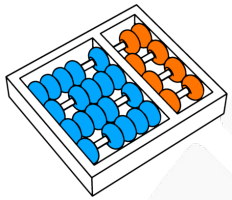
$$\max \quad 50x_p - 20x_s$$

# Programação linear inteira mista



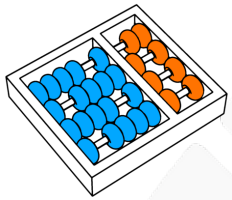
# Programação linear inteira mista

$$\begin{aligned} \min \quad & c^t \mathbf{x} + d^t \mathbf{y} \\ \text{s.a:} \quad & \\ & A\mathbf{x} + B\mathbf{y} \geq b \\ & \mathbf{x} \in \mathbb{Q}_+^{n_1}, \mathbf{y} \in \mathbb{Z}_+^{n_2} \end{aligned}$$



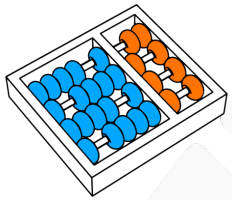
# Programação linear inteira pura

$$\begin{aligned} \min \quad & c^t \mathbf{x} \\ \text{s.a:} \quad & \\ & A\mathbf{x} \geq b \\ & \mathbf{x} \in \mathbb{Z}_+^n \end{aligned}$$



# Programação linear binária (ou 0-1)

$$
\begin{aligned}
\min \quad & c^t \mathbf{x} \\
\text{s.a:} \quad & \\
& A\mathbf{x} \geq b \\
& \mathbf{x} \in \{0,1\}_+^n
\end{aligned}
$$



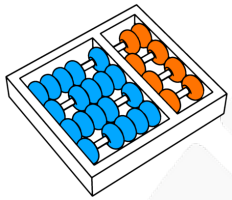
## Algumas equivalências

$$\min\ c^t\mathbf{x} \equiv \max\ -c^t\mathbf{x}$$

$$A\mathbf{x} \geq b \equiv -A\mathbf{x} \leq -b$$

$$A\mathbf{x} = b \equiv \begin{cases} A\mathbf{x} \leq b \\ A\mathbf{x} \geq b \end{cases}$$

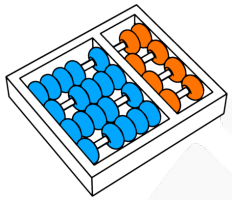
$$\mathbf{x_i} \in \mathbb{Z} \equiv \begin{cases} \mathbf{x_i'}, \mathbf{x_i''} \in \mathbb{Z}_+ \\ \mathbf{x_i} = \mathbf{x_i'} - \mathbf{x_i''} \end{cases}$$



## Vantagens

Melhora consideravelmente a capacidade de modelar:

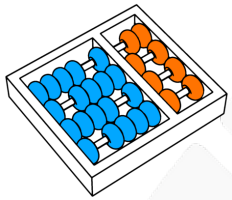
- Maior flexibilidade na modelagem.
- Modelos mais realistas.
- Permite modelar restrições lógicas.
- Capaz de modelar funções não lineares.



## Desvantagens

- Maior dificuldade para modelar.
- Pode ser muito mais difícil de solucionar.

# Dicas para formular



# Variáveis binárias

**VARIÁVEIS BINÁRIAS** só tomam valor em $\{0, 1\}$ e permitem modelar diferentes cenários:

- Se uma solução pode ou não satisfazer algumas propriedades (ou condições), podemos definir para cada propriedade a variável:

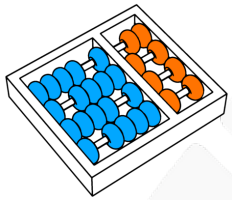
  $$x_p = \begin{cases} 1 \text{ se a solução satisfaz } p \\ 0, \text{ em caso contrário} \end{cases}$$

- Se uma solução requer selecionar elementos em um conjunto, podemos definir, para cada elemento $e$ do conjunto a variável:

  $$x_e = \begin{cases} 1 \text{ se a solução seleciona o elemento } p \\ 0, \text{ em caso contrário} \end{cases}$$

- Se uma solução precisa ordenar elementos de um conjunto, para cada par de elementos $i$ e $j$ podemos definir: $x_{ij} = \begin{cases} 1, \text{ se } i \text{ está antes que } j \text{ na solução} \\ 0, \text{ em caso contrário} \end{cases}$

- Se uma solução precisa associar pares de elementos, para cada par de elementos $i$ e $j$ podemos definir: $x_{ij} = \begin{cases} 1, \text{ se } i \text{ está associado com } j \text{ na solução} \\ 0, \text{ em caso contrário} \end{cases}$

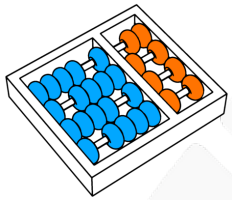


# Restrições lógicas com variáveis binárias

**NOT**: Desejamos garantir que $z$ é $1$ se e somente se $x$ é $0$:

$$\mathbf{z} = 1 - \mathbf{x}$$

| $z = \neg x$ | $x$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

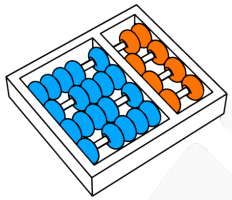


# Restrições lógicas com variáveis binárias

**AND**: Desejamos garantir que $z$ é $1$ se e somente se $x$ e $y$ são $1$:

$$\begin{aligned} 2 \cdot \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ \mathbf{z} + 1 &\geq \mathbf{x} + \mathbf{y} \end{aligned}$$

| $z = x \wedge y$ | $x$ | $y$ |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |



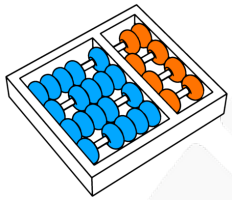
# Restrições lógicas com variáveis binárias

**OR**: Desejamos garantir que $z$ é $1$ se e somente se $x$ ou $y$ são $1$:

$$\begin{aligned} \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\ 2 \cdot \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \end{aligned}$$

| $z = x \vee y$ | $x$ | $y$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 0 | 0 |

Para obrigar $x \vee y$ podemos usar $x + y \geq 1$.



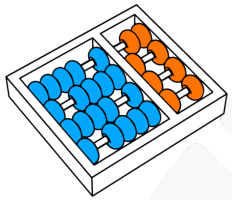
# Restrições lógicas com variáveis binárias

**XOR**: Desejamos garantir que $z$ é $1$ se e somente se exatamente uma entre $x$ e $y$ é $1$:

$$\begin{aligned} z &\leq x + y \\ 2 - z &\geq x + y \\ z &\geq x - y \\ z &\geq y - x \end{aligned}$$

| $z = x \oplus y$ | $x$ | $y$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 0 | 0 |

Para obrigar $x \oplus y$ podemos usar $x + y = 1$.



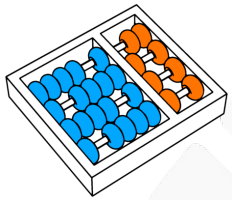
# Restrições lógicas com variáveis binárias

**EQUIVALÊNCIA**: Desejamos garantir que $z$ é $1$ se e somente se $x$ é igual a $y$:

$$
\begin{aligned}
1 - \mathbf{z} &\leq \mathbf{x} + \mathbf{y} \\
1 + \mathbf{z} &\geq \mathbf{x} + \mathbf{y} \\
1 - \mathbf{z} &\geq \mathbf{x} - \mathbf{y} \\
1 - \mathbf{z} &\geq \mathbf{y} - \mathbf{x}
\end{aligned}
$$

| $z = x \Leftrightarrow y$ | $x$ | $y$ |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |

Para obrigar $x \Leftrightarrow y$ podemos usar $x - y = 0$.



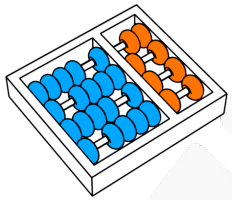
# Restrições lógicas com variáveis binárias

**IMPLICAÇÃO**: Desejamos garantir que $z$ é $1$ se e somente se $y$ é $1$ sempre que $x$ for $1$:

$$\begin{aligned} \mathbf{z} - 1 &\leq \mathbf{y} - \mathbf{x} \\ 2 \cdot \mathbf{z} - 1 &\geq \mathbf{y} - \mathbf{x} \end{aligned}$$

| $z = x \Rightarrow y$ | $x$ | $y$ |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 0 | 0 |

Para obrigar $x \Rightarrow y$ podemos usar $y \geq x$.



## $M$ grande para associar condições a variáveis binárias

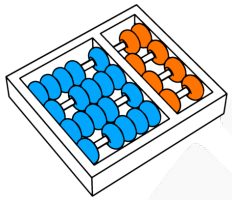
Suponha que temos uma condição (propriedade) simples escrita como:

$$a^T \cdot \mathbf{x} \leq b$$

Para associar uma variável binária com essa condição, encontramos um valor **M grande o suficiente**, tal que $a^T \cdot \mathbf{x} \leq b + \mathbf{M}$ para qualquer solução viável $\mathbf{x}$.

Então, definimos a variável binária $\mathbf{y}$ e escrevemos a restrição:

$$a^T \cdot \mathbf{x} \leq b + \mathbf{M} \cdot (1 - \mathbf{y})$$



## Associando condições com variáveis binárias

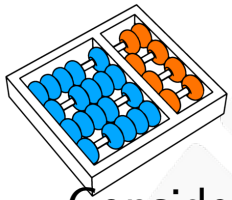
Considere o conjunto $\{c_i\}_{i=1}^{n}$ de condições (ou propriedades) já associada cada uma com uma variável binária $\{\mathbf{x_i}\}_{i=1}^{n}$:

- Para indicar se pelo menos $k$ condições são satisfeitas, podemos definir a variável binária $z$ e as restrições:

$$k - 1 + n \cdot \mathbf{z} \geq \sum_{i=1}^{n} \mathbf{x_i} \quad \text{e} \quad k - n \cdot (1 - \mathbf{z}) \leq \sum_{i=1}^{n} \mathbf{x_i}$$

- Para indicar se no máximo $k$ condições são satisfeitas, podemos definir a variável binária $z$ e as restrições:

$$k + 1 - n \cdot \mathbf{z} \leq \sum_{i=1}^{n} \mathbf{x_i} \quad \text{e} \quad k + n \cdot (1 - \mathbf{z}) \geq \sum_{i=1}^{n} \mathbf{x_i}$$



# Restrições para seleção de elementos

Considere um conjunto de $n$ elementos para selecionar (podem ser condições ou propriedades também), onde cada um é associado com uma variável binária $\{x_i\}_{i=1}^{n}$:

- Para selecionar (ou satisfazer) pelo menos $k$ elementos (ou condições), podemos definir:

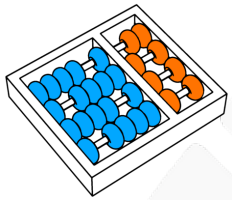
$$\sum_{i=1}^{n} x_i \geq k$$

- Para selecionar (ou satisfazer) exatamente $k$ elementos (ou condições), podemos definir:

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

- Para selecionar (ou satisfazer) no máximo $k$ elementos (ou condições), podemos definir:

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



# Variáveis e restrições para seleção de intervalos

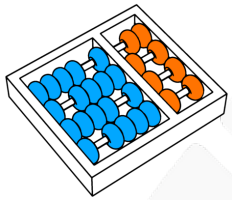
Considere a seguinte expressão em que cada $h_i$ é uma função linear sem termo independente e cada $c_i$ é uma constante:

$$f(\mathbf{x}) = \begin{cases} c_1 + h_1(\mathbf{x}) & \text{, se } \ell_1 \leq \mathbf{x} \leq \nu_1 \\ c_2 + h_2(\mathbf{x}) & \text{, se } \ell_2 \leq \mathbf{x} \leq \nu_2 \\ \dots & \\ c_n + h_n(\mathbf{x}) & \text{, se } \ell_n \leq \mathbf{x} \leq \nu_n \end{cases}$$

Podemos definir as variáveis binárias $\mathbf{y_i} = \begin{cases} 1, \text{ se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, \text{ em caso contrário} \end{cases}$ e as variáveis $\mathbf{z_i} = \begin{cases} \mathbf{x}, \text{ se } \ell_i \leq \mathbf{x} \leq \nu_i \\ 0, \text{ em caso contrário} \end{cases}$, escrevendo a expressão como:

$$f(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^{n} c_i \cdot \mathbf{y_i} + h_i(\mathbf{z_i})$$

Com as restrições: $\ell_i \cdot \mathbf{y_i} \leq z_i \leq \nu_i \cdot \mathbf{y_i}$ (para cada $i$) e $\sum_{i=1}^{n} \mathbf{y_i} = 1$.



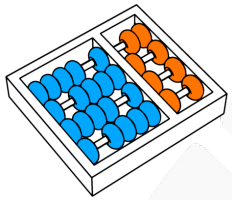
## Restrições para seleção condicional

Dadas três condições (ou elementos) cada uma associada com variáveis binárias $x$, $y$ e $z$. Para garantir que $z$ é $1$ sempre que $x$ e $y$ são $1$ (ou seja, **se** $x = 1$ **e** $y = 1$**, então** $z = 1$), podemos definir a restrição:

$$x + y \leq 1 + z$$

Generalizando a um conjunto de $n + 1$ variáveis binárias, a condição **se** $x_1 = 1$ **e** $x_2 = 1$ **e** $\ldots$ **e** $x_n = 1$**, então** $x_{n+1} = 1$, pode ser formulada como:

$$\sum_{i=1}^{n} x_i \leq n - 1 + x_{n+1}$$

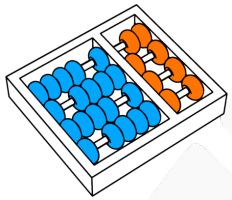


## Restrições para ordem

Se desejamos representar uma ordenação de $n$ elementos, podemos definir para cada par $(i, j)$ de elementos a variável binária $\mathbf{x_{ij}}$ para expressar se o elemento $i$ está na frente do $j$:

$$\mathbf{x_{ij}} + \mathbf{x_{ji}} = 1 \quad \text{para cada par } (i, j) \text{ (antissimetria)}$$

$$\mathbf{x_{ij}} + \mathbf{x_{jk}} \leq 1 + \mathbf{x_{ik}} \quad \text{para cada três elementos } (i, j, k) \text{ (transitividade)}$$



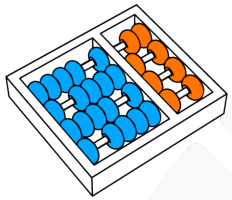
## Restrições para ordem

Dados $n$ elementos, existem $n$ posições para ordená-los. Logo, podemos definir a variável binária $\mathbf{x_{ij}}$ para indicar se o elemento $i$ está na posição $j$:

$$\sum_{i=1}^{n} \mathbf{x_{ij}} = 1 \qquad \text{para cada posição } j \text{ (um elemento por posição)}$$

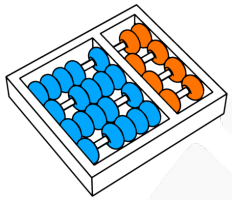
$$\sum_{j=1}^{n} \mathbf{x_{ij}} = 1 \qquad \text{para cada elemento } i \text{ (uma posição para cada elemento)}$$

# Exemplos



## Mochila binária

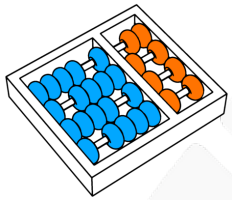
Dados uma mochila com capacidade máxima de peso $W$ e um conjunto finito de objetos $O$, cada um com um peso ($\omega$) e um valor ($\nu$) associados: $\omega_o$ e $nu_o$ para cada $o \in O$.
O problema procura por uma combinação de objetos para carregar na mochila, de forma que a soma dos pesos não ultrapasse o peso da mochila e a soma dos valores seja maximizada.



# Mochila binária

Devemos decidir se cada objeto é selecionado ou não:

$$x_o = \begin{cases} 1 & \text{, se o objeto } o \text{ é selecionado} \\ 0 & \text{, em caso contrário} \end{cases}$$



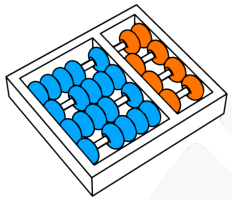
## Mochila binária

O objetivo é maximizar a soma dos valores selecionados:

$$\max \quad \sum_{o \in O} \nu_o \cdot \mathbf{x_o}$$

A soma dos pesos selecionados não deve ultrapassar o peso da mochila:

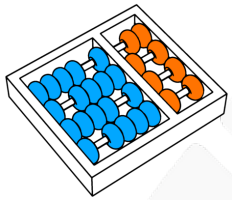
$$\sum_{o \in O} \omega_o \cdot \mathbf{x_o} \leq W$$



# Mochila binária

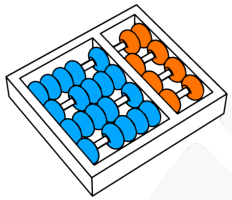
**Instância**: $O$, $\nu : O \to \mathbb{Q}_+$, $\omega : O \to \mathbb{Q}_+$ e $W \in \mathbb{Q}_+$.

$$\begin{aligned} \max \quad & \sum_{o \in O} \nu_o \cdot \mathbf{x_o} \\ s.a. : \quad & \\ & \sum_{o \in O} \omega_o \cdot \mathbf{x_o} \leq W \\ & \mathbf{x_o} \in \{0,1\} \quad \forall o \in O \end{aligned}$$



## Particionamento balanceado

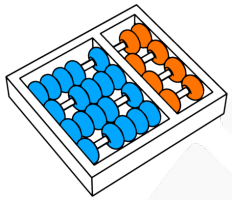
Dado um conjunto finito de elementos $E$ com valores positivos ($v$) associados ($v_i$ para cada $i \in E$), se procura uma partição de $E$ em dois conjuntos $U$ e $V$ que minimize $\left|\sum_{i \in U} v_i - \sum_{i \in E \setminus U} v_i\right|$.



# Particionamento balanceado

Por cada elemento de $E$, devemos decidir se está no primeiro conjunto da partição ($U$) ou no segundo ($E \setminus U$):

$$x_i = \begin{cases} 1 & \text{, se } i \text{ está em } U \\ 0 & \text{, em caso contrário} \end{cases}$$



## Partição balanceada

Função objetivo não linear:

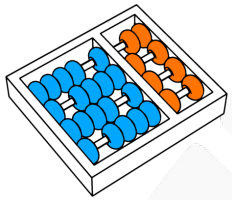
$$\min \quad \left| \sum_{i \in E} v_i \cdot \mathbf{x_i} - \sum_{i \in E} v_i \cdot (1 - \mathbf{x_i}) \right|$$

Esse objetivo é equivalente a minimizar o máximo entre $\sum_{i \in E} v_i \cdot \mathbf{x_i} - \sum_{i \in E} v_i \cdot (1 - \mathbf{x_i})$ e $\sum_{i \in E} v_i \cdot (1 - \mathbf{x_i}) - \sum_{i \in E} v_i \cdot \mathbf{x_i}$.

Podemos alcançar isso minimizando a variável $\mathbf{z} \in \mathbb{Z}_+$, onde:

$$\sum_{i \in E} v_i \cdot \mathbf{x_i} - \sum_{i \in E} v_i \cdot (1 - \mathbf{x_i}) \leq \mathbf{z}$$

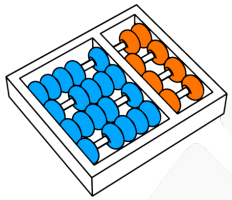
$$\sum_{i \in E} v_i \cdot (1 - \mathbf{x_i}) - \sum_{i \in E} v_i \cdot \mathbf{x_i} \leq \mathbf{z}$$



# Partição balanceada

**Instância**: $E$, $v : E \to \mathbb{Z}_+$.

$$\begin{aligned} \min \quad & \mathbf{z} \\ s.a. : \quad & \\ & \sum_{i \in E} v_i \cdot \mathbf{x_i} - \sum_{i \in E} v_i \cdot (1 - \mathbf{x_i}) \leq \mathbf{z} \\ & \sum_{i \in E} v_i \cdot (1 - \mathbf{x_i}) - \sum_{i \in E} v_i \cdot \mathbf{x_v} \leq \mathbf{z} \\ & \mathbf{z} \in \mathbb{Z}_+, \mathbf{x_i} \in \{0, 1\} \quad \forall i \in E \end{aligned}$$

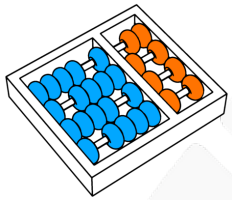


## Partição balanceada e mochila

O mínimo entre $\sum_{i \in E} v_i \cdot \mathbf{x_i}$ e $\sum_{i \in E} v_i \cdot (1 - \mathbf{x_i})$ é no máximo $\frac{1}{2} \sum_{i \in E} v_i$.

Portanto, outra opção para evitar o valor absoluto seria garantir que a soma dos elementos em $U$ não ultrapasse o valor anterior:

$$\sum_{i \in E} v_i \cdot \mathbf{x_i} \leq \frac{1}{2} \sum_{i \in E} v_i$$



# Partição balanceada e mochila

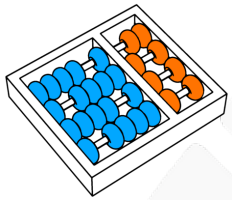
**Instância**: $E$, $v : E \to \mathbb{Z}_+$.

$$\min \quad \sum_{i \in E} v_i \cdot (1 - \mathbf{x_i}) - \sum_{i \in E} v_i \cdot \mathbf{x_i}$$

$$s.a. :$$

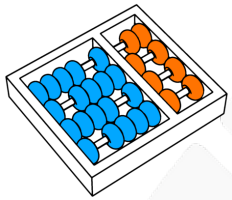
$$\sum_{i \in E} v_i \cdot \mathbf{x_i} \leq \frac{1}{2} \sum_{i \in E} v_i$$

$$\mathbf{x_i} \in \{0, 1\} \quad \forall i \in E$$



## Escalonamento de tarefas

Dadas $m$ máquinas iguais e $n$ tarefas, onde a tarefa $j$ requer $t_j \in \mathbb{Z}_+$ tempo para ser processada, se deseja atribuir cada tarefa a uma máquina de forma a minimizar o tempo de processamento na máquina em que as tarefas atribuídas somem o maior tempo.

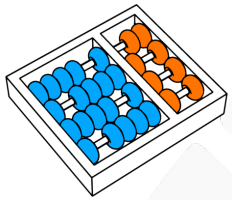


# Escalonamento de tarefas

Para cada par tarefa-máquina, podemos definir uma variável binária que indique se a tarefa foi ou não atribuída à máquina:

$$x_{ij} = \begin{cases} 1 & \text{, se a tarefa } j \text{ foi atribuída à máquina } i \\ 0 & \text{, em caso contrário} \end{cases}$$

Também podemos definir uma variável inteira $z$ que indique o maior tempo de processamento entre as máquinas.



## Escalonamento de tarefas

O objetivo é minimizar o valor de $\mathbf{z}$:

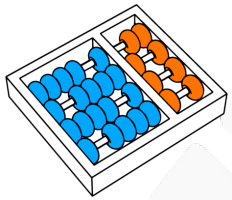
$$\min \quad \mathbf{z}$$

Como $\mathbf{z}$ é o maior o tempo de processamento das máquinas, o valor deve limitar o tempo de processamento de qualquer máquina:

$$\sum_{j=1}^{n} t_j \cdot \mathbf{x_{ij}} \leq \mathbf{z} \qquad \forall 1 \leq i \leq m$$

Cada tarefa deve ser atribuída exatamente a uma máquina:

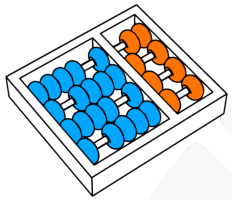
$$\sum_{i=1}^{m} \mathbf{x_{ij}} = 1 \qquad \forall 1 \leq j \leq n$$



# Escalonamento de tarefas

**Instância**: $m$, $n$ e $\{t_j\}_{j=1}^{n}$.

$$\begin{aligned} \min \quad & \mathbf{z} \\ s.a.: \quad & \\ & \textstyle\sum_{j=1}^{n} t_j \cdot x_{ij} \leq \mathbf{z} && \forall 1 \leq i \leq m \\ & \textstyle\sum_{i=1}^{m} \mathbf{x_{ij}} = 1 && \forall 1 \leq j \leq n \\ & \mathbf{z} \in \mathbb{Z}_+, \mathbf{x_{ij}} \in \{0,1\} && \forall 1 \leq i \leq m, 1 \leq j \leq n \end{aligned}$$

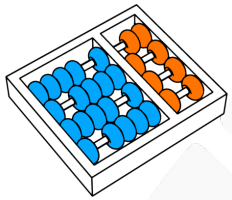


## Clique e conjunto independente

Um **CONJUNTO INDEPENDENTE** em um grafo, é um conjunto de vértices que não contém adjacentes.

Uma **CLIQUE** é um subgrafo onde todo par de vértices são adjacentes.

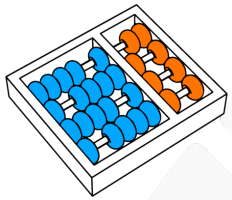
Formulações para o conjunto independe máximo e para a clique máxima podem ser muito similares.



## Clique e conjunto independente

Para cada vértice precisamos decidir se ele está ou não em uma solução (clique ou conjunto independente):

$$x_v = \begin{cases} 1 & \text{, se o vértice } v \text{ é selecionado para a solução} \\ 0 & \text{, em caso contrário} \end{cases}$$



## Clique e conjunto independente

O objetivo pode ser visto como maximizar o número de vértices selecionados na solução:

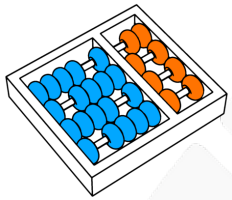
$$\max \quad \sum_{v \in V} x_v$$

No conjunto independente, dois vértices selecionados não podem ser adjacentes. Ou seja, por cada aresta $(u, v)$ no máximo podemos selecionar só um dos extremos:

$$x_u + x_v \leq 1 \qquad \forall (u, v) \in E$$

Na clique, todos os vértices devem ser adjacentes. Ou seja, por cada par de vértices **não** adjacentes (arestas do complemento) podemos selecionar no máximo um:

$$x_u + x_v \leq 1 \qquad \forall (u, v) \notin E$$



# Clique e conjunto independente

Conjunto independente máximo
**Instância**: $G = (V, E)$.

$$\max \quad \sum_{v \in V} x_v$$

$$s.a.:$$

$$x_u + x_v \leq 1 \quad \forall (u, v) \in E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

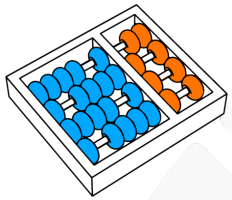
Clique máxima
**Instância**: $G = (V, E)$.

$$\max \quad \sum_{v \in V} x_v$$

$$s.a.:$$

$$x_u + x_v \leq 1 \quad \forall (u, v) \notin E$$

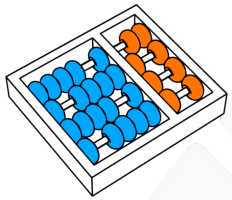
$$x_v \in \{0, 1\} \quad \forall v \in V$$



## Coloração de vértices

Uma **COLORAÇÃO DE VÉRTICES** em um grafo é uma atribuição de cor a cada vértices de forma que não hajam dois vértices adjacentes com a mesma cor.

A coloração de vértices mínima, procura por uma coloração que minimiza o número de cores usadas.



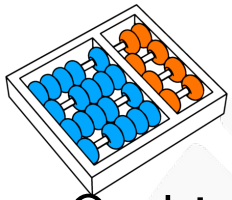
## Coloração de vértices

No máximo há $|V|$ cores, uma por cada vértice, portanto podemos definir variáveis binárias por cada possível cor, que indique se ela é usada ou não:

$$x_c = \begin{cases} 1 & \text{, se a cor } c \text{ é usada} \\ 0 & \text{, em caso contrário} \end{cases}$$

Para identificar a cor usada em cada vértice, podemos definir as seguintes variáveis binárias:

$$y_{vc} = \begin{cases} 1 & \text{, se o vértice } v \text{ usa a cor } c \\ 0 & \text{, em caso contrário} \end{cases}$$



## Coloração de vértices

O objetivo é minimizar o número de cores usadas:

$$\min \quad \sum_{c \in C} x_c$$

Cada vértice deve ter exatamente uma cor:

$$\sum_{c \in C} y_{vc} = 1 \quad \forall v \in V$$

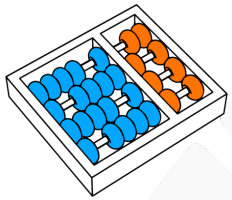
Vértices adjacentes não podem ter a mesma cor:

$$y_{uc} + y_{vc} \leq 1 \quad \forall c \in C, (u, v) \in E$$

Devemos associar as cores dos vértices com as cores usadas:

$$x_c \geq y_{vc} \qquad \forall c \in C, v \in V$$

$$x_c \leq \sum_{v \in V} y_{vc} \qquad \forall c \in C$$



## Coloração de vértices

**Instância**: $G = (V, E)$.
Defina $C = \{1, 2, \ldots, |V|\}$ (possíveis cores)

$$\begin{aligned}
\min \quad & \sum_{c \in C} \mathbf{x_c} \\
s.a.: \quad & \\
& \sum_{c \in C} \mathbf{y_{vc}} = 1 && \forall v \in V \\
& \mathbf{y_{uc}} + \mathbf{y_{vc}} \leq 1 && \forall c \in C, (u, v) \in E \\
& \mathbf{x_c} \geq \mathbf{y_{vc}} && \forall c \in C, v \in V \\
& \mathbf{x_c} \leq \sum_{v \in V} \mathbf{y_{vc}} && \forall c \in C \\
& \mathbf{x_c}, \mathbf{y_{vc}} \in \{0, 1\} && \forall c \in C, v \in V
\end{aligned}$$

# Programação linear inteira

MC558 - Projeto e Análise de Algoritmos II

11/24

22

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP