

# Projeto e Análise de Algoritmos II (MC558)

## Introdução à Programação Linear

Prof. Dr. Ruben Interian

# Resumo

- 1 Revisão do conteúdo e objetivo
- 2 Programação Linear
- 3 Síntese

# Resumo

- 1 Revisão do conteúdo e objetivo
- 2 Programação Linear
- 3 Síntese

# Revisão do conteúdo

- Diversos problemas podem ser modelados usando formulações de **Programação Linear** e **Programação Linear Inteira**.
- Criar uma formulação de **PL** ou **PLI** do problema **P** é a mesma coisa que criar uma redução  $P \propto PL$  ou  $P \propto PLI$ .

# Objetivo

Responder as perguntas:

- Sempre existe **solução viável** da minha formulação de um problema de Programação Linear?
- Sempre existe **solução ótima** da minha formulação de um problema de Programação Linear?

Veremos mais alguns exemplos de reduções para **PL** e **PLI**.

# Resumo

- 1 Revisão do conteúdo e objetivo
- 2 Programação Linear
- 3 Síntese



# PL e PLI – soluções viáveis e soluções ótimas

Em um problema de **PL** ou **PLI**:

- Uma **solução viável** é aquela que satisfaz as restrições.



# PL e PLI – soluções viáveis e soluções ótimas

Em um problema de **PL** ou **PLI**:

- Uma **solução viável** é aquela que satisfaz as restrições.
- Todas as soluções viáveis formam a **região viável**.

# PL e PLI – soluções viáveis e soluções ótimas

Em um problema de **PL** ou **PLI**:

- Uma **solução viável** é aquela que satisfaz as restrições.
- Todas as soluções viáveis formam a **região viável**.
- Uma **solução ótima** é a melhor solução viável do problema, dada a função objetivo.

## PL e PLI – soluções viáveis e soluções ótimas

Em um problema de **PL** ou **PLI**:

- Uma **solução viável** é aquela que satisfaz as restrições.
- Todas as soluções viáveis formam a **região viável**.
- Uma **solução ótima** é a melhor solução viável do problema, dada a função objetivo.

### Perguntas:

- Os problemas **PL** e **PLI** sempre possuem uma **solução viável**?

## PL e PLI – soluções viáveis e soluções ótimas

Em um problema de **PL** ou **PLI**:

- Uma **solução viável** é aquela que satisfaz as restrições.
- Todas as soluções viáveis formam a **região viável**.
- Uma **solução ótima** é a melhor solução viável do problema, dada a função objetivo.

### Perguntas:

- Os problemas **PL** e **PLI** sempre possuem uma **solução viável**?
- Se há alguma **solução viável**, sempre existe uma **solução ótima**?

# PL e PLI – soluções viáveis e soluções ótimas

Em um problema de **PL** ou **PLI**:

- Uma **solução viável** é aquela que satisfaz as restrições.
- Todas as soluções viáveis formam a **região viável**.
- Uma **solução ótima** é a melhor solução viável do problema, dada a função objetivo.

**Perguntas:**

- Os problemas **PL** e **PLI** sempre possuem uma **solução viável**?
- Se há alguma **solução viável**, sempre existe uma **solução ótima**?
- Dado um algoritmo para resolver **PL** / **PLI**, que tipo de saída ele oferece?

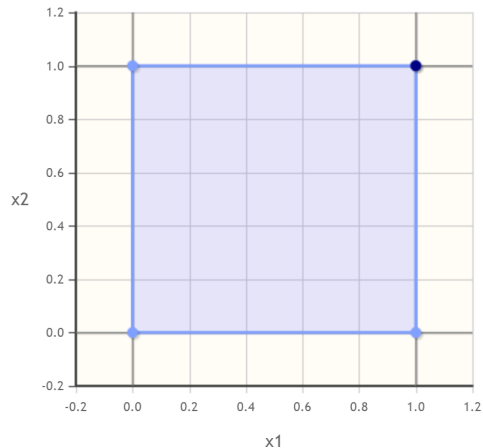
## PL e PLI – soluções viáveis e soluções ótimas

### Caso 1: Problema viável e limitado.

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{sujeito a} \quad & x_1 \leq 1 \\ & x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

### Caso 1: Problema **viável** e **limitado**.

No caso do problema de **PL**, sempre **há solução ótima**, e uma solução ótima é sempre **um extremo da região viável**.



# PL e PLI – soluções viáveis e soluções ótimas

**Caso 2:** Problema **viável** e **ilimitado**.

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{sujeito a} \quad & x_1 - x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$



## PL e PLI – soluções viáveis e soluções ótimas

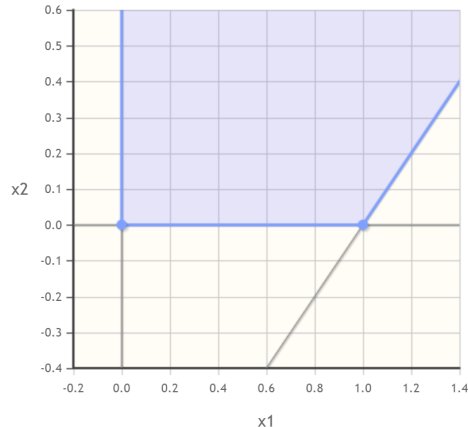
### Caso 2: Problema **viável** e **ilimitado**.

$$\max x_1 + x_2$$

sujeito a  $x_1 - x_2 \leq 1$

$$x_1, x_2 \geq 0$$

## Podemos não ter ótimo ...



# PL e PLI – soluções viáveis e soluções ótimas

**Caso 3:** Problema **inviável**  
(não há nenhuma solução viável).

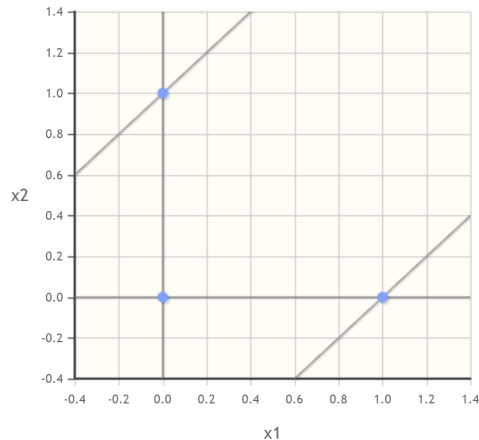
$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{sujeito a} \quad & x_1 - x_2 \geq 1 \\ & x_1 - x_2 \leq -1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

# PL e PLI – soluções viáveis e soluções ótimas

**Caso 3:** Problema **inviável**  
(não há nenhuma solução viável).

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{sujeito a} \quad & x_1 - x_2 \geq 1 \\ & x_1 - x_2 \leq -1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Neste caso, certamente não há ótimo.



# PL e PLI – soluções viáveis e soluções ótimas

## Respostas:

- Os problemas **PL** e **PLI** sempre possuem uma **solução viável**?  
**Não**. O problema pode ser **inviável**.

# PL e PLI – soluções viáveis e soluções ótimas

## Respostas:

- Os problemas **PL** e **PLI** sempre possuem uma **solução viável**?  
**Não**. O problema pode ser **inviável**.
- Se há alguma **solução viável**, sempre existe uma **solução ótima**?  
**Não**. O problema pode ser **viável** e **ilimitado**.

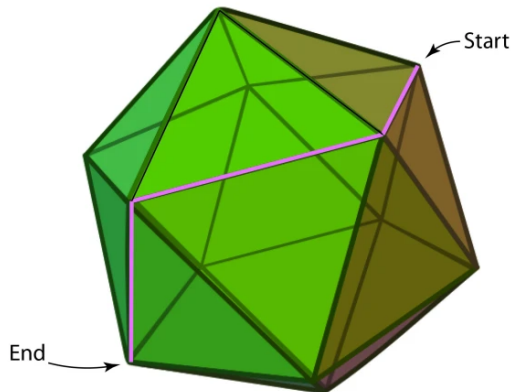
# PL e PLI – soluções viáveis e soluções ótimas

## Respostas:

- Os problemas **PL** e **PLI** sempre possuem uma **solução viável**?  
**Não**. O problema pode ser **inviável**.
- Se há alguma **solução viável**, sempre existe uma **solução ótima**?  
**Não**. O problema pode ser **viável** e **ilimitado**.
- Se o problema é **viável** e **limitado** – temos **solução viável** e **solução ótima**.

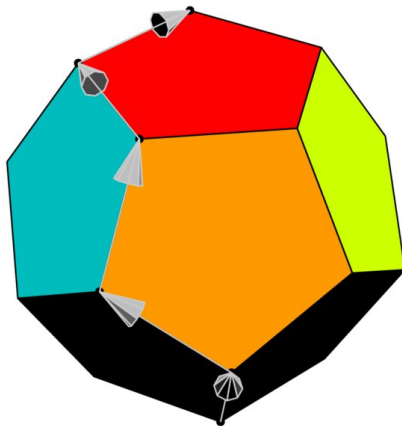
# PL e PLI – busca de soluções ótimas

**Simplex:** Primeiro algoritmo para **PL**. Muito **eficiente** na prática, (embora ineficiente em casos raros), usado e ensinado até hoje.



# PL e PLI – busca de soluções ótimas

**Simplex:** Primeiro algoritmo para **PL**. Muito **eficiente** na prática, (embora ineficiente em casos raros), usado e ensinado até hoje.





# PL e PLI – busca de soluções ótimas

O Simplex foi proposto pelo matemático norte-americano [George B. Dantzig](#) em 1947.

# PL e PLI – busca de soluções ótimas

O Simplex foi proposto pelo matemático norte-americano [George B. Dantzig](#) em 1947.

*Dantzig era um estudante de doutorado em Berkley orientado pelo prof. J. Neyman. Dantzig chegou atrasado um dia para uma das aulas de Neyman. No quadro ele encontrou dois problemas e assumiu que foram deixados como tarefa de casa. Alguns dias depois, ele pediu desculpas a Neyman por demorar tanto para fazer a tarefa: “os problemas pareciam estar **um pouco mais difíceis do que o normal**”.*

# PL e PLI – busca de soluções ótimas

O Simplex foi proposto pelo matemático norte-americano [George B. Dantzig](#) em 1947.

*Dantzig era um estudante de doutorado em Berkley orientado pelo prof. J. Neyman. Dantzig chegou atrasado um dia para uma das aulas de Neyman. No quadro ele encontrou dois problemas e assumiu que foram deixados como tarefa de casa. Alguns dias depois, ele pediu desculpas a Neyman por demorar tanto para fazer a tarefa: “os problemas pareciam estar **um pouco mais difíceis do que o normal**”.*

*Seis semanas depois, Neyman disse a Dantzig que tinha acabado de escrever um artigo. “Leia-o para que eu possa enviá-lo para publicação.” Dantzig não tinha ideia do que Neyman estava falando até ele explicar que os dois problemas nos quadros eram importantes **problemas não resolvidos** de estatística – não eram dever de casa.*

# PL e PLI – busca de soluções ótimas

O Simplex foi proposto pelo matemático norte-americano [George B. Dantzig](#) em 1947.

*Dantzig era um estudante de doutorado em Berkley orientado pelo prof. J. Neyman. Dantzig chegou atrasado um dia para uma das aulas de Neyman. No quadro ele encontrou dois problemas e assumiu que foram deixados como tarefa de casa. Alguns dias depois, ele pediu desculpas a Neyman por demorar tanto para fazer a tarefa: “os problemas pareciam estar **um pouco mais difíceis do que o normal**”.*

*Seis semanas depois, Neyman disse a Dantzig que tinha acabado de escrever um artigo. “Leia-o para que eu possa enviá-lo para publicação.” Dantzig não tinha ideia do que Neyman estava falando até ele explicar que os dois problemas nos quadros eram importantes **problemas não resolvidos** de estatística – não eram dever de casa.*

*Durante a sua pós-graduação, Dantzig teve dificuldade em encontrar um tema para sua tese. Quando Dantzig contou isso a Neyman, o professor lhe disse apenas para **colocar os dois problemas numa pasta** – ele os aceitaria como a sua tese.*

# PLI: Exemplo 1 (Problema da Mochila)

## Problema da Mochila (Knapsack problem)

Numa caverna, você encontra um tesouro formado por  $n$  itens distintos. Você possui uma mochila que pode carregar uma capacidade total de  $W$  kg. Cada um dos  $n$  itens possui um peso  $w_i$ , e um valor  $v_i$ , para  $i = 1, \dots, n$ . Queremos encontrar um conjunto de itens com o maior valor total possível, que não ultrapasse a capacidade da mochila.

Em resumo, precisamos encher a mochila de itens, sem ultrapassar um determinado limite de peso, otimizando o valor total dos itens escolhidos.

# PLI: Exemplo 1 (Problema da Mochila)

- **Variáveis:** Para cada  $i = 1, \dots, n$ , defina uma variável  $x_i$ :

$$x_i = \begin{cases} 1 & \text{indica que o item } i \text{ será escolhido,} \\ 0 & \text{indica que o item } i \text{ não será escolhido.} \end{cases}$$

# PLI: Exemplo 1 (Problema da Mochila)

- **Variáveis:** Para cada  $i = 1, \dots, n$ , defina uma variável  $x_i$ :

$$x_i = \begin{cases} 1 & \text{indica que o item } i \text{ será escolhido,} \\ 0 & \text{indica que o item } i \text{ não será escolhido.} \end{cases}$$

- **Função objetivo:**  $\sum_{i=1}^n v_i x_i$

# PLI: Exemplo 1 (Problema da Mochila)

- **Variáveis:** Para cada  $i = 1, \dots, n$ , defina uma variável  $x_i$ :

$$x_i = \begin{cases} 1 & \text{indica que o item } i \text{ será escolhido,} \\ 0 & \text{indica que o item } i \text{ não será escolhido.} \end{cases}$$

- **Função objetivo:**  $\sum_{i=1}^n v_i x_i$
- **Restrições:**  $\sum_{i=1}^n w_i x_i \leq W$



# PLI: Exemplo 1 (Problema da Mochila)

- **Variáveis:** Para cada  $i = 1, \dots, n$ , defina uma variável  $x_i$ :

$$x_i = \begin{cases} 1 & \text{indica que o item } i \text{ será escolhido,} \\ 0 & \text{indica que o item } i \text{ não será escolhido.} \end{cases}$$

- **Função objetivo:**  $\sum_{i=1}^n v_i x_i$
- **Restrições:**  $\sum_{i=1}^n w_i x_i \leq W$
- **Não-negatividade:**  $x_i \geq 0$ , para cada  $i = 1, \dots, n$ .

# PLI: Exemplo 1 (Problema da Mochila)

- **Variáveis:** Para cada  $i = 1, \dots, n$ , defina uma variável  $x_i$ :

$$x_i = \begin{cases} 1 & \text{indica que o item } i \text{ será escolhido,} \\ 0 & \text{indica que o item } i \text{ não será escolhido.} \end{cases}$$

- **Função objetivo:**  $\sum_{i=1}^n v_i x_i$
- **Restrições:**  $\sum_{i=1}^n w_i x_i \leq W$
- **Não-negatividade:**  $x_i \geq 0$ , para cada  $i = 1, \dots, n$ .
- **Integralidade:**  $x_i \in \{0, 1\}$ , para cada  $i = 1, \dots, n$ .

# PLI: Exemplo 1 (Problema da Mochila)

**Formulação de PLI para o Problema da Mochila:**

$$\begin{aligned} \max \quad & \sum_{i=1}^n v_i x_i \\ \text{sujeito a} \quad & \sum_{i=1}^n w_i x_i \leq W \\ & x_i \in \{0, 1\} \text{ para cada } i = 1, \dots, n \end{aligned}$$

# PLI: Exemplo 1 (Problema da Mochila)

Formulação de PLI para o Problema da Mochila:

$$\begin{aligned} \max \quad & \sum_{i=1}^n v_i x_i \\ \text{sujeito a} \quad & \sum_{i=1}^n w_i x_i \leq W \\ & x_i \in \{0, 1\} \text{ para cada } i = 1, \dots, n \end{aligned}$$

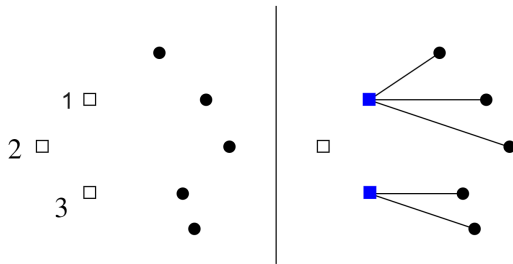
A formulação possui  $n$  variáveis e apenas 1 restrição!

## PLI: Exemplo 2 (Facility Location)

Uma empresa conhece a localização de seus  $n$  clientes denotados por  $j = 1, 2, \dots, n$ . Para atendê-los a companhia decide abrir um conjunto de instalações: cada cliente será atendido por exatamente uma instalação. Uma instalação pode atender vários clientes. As possíveis instalações são denotadas por  $i = 1, 2, \dots, m$ . A empresa conhece o custo de abertura  $f_i$  da instalação  $i$ , e o custo de envio  $c_{ij}$  da instalação  $i$  a um cliente  $j$ .

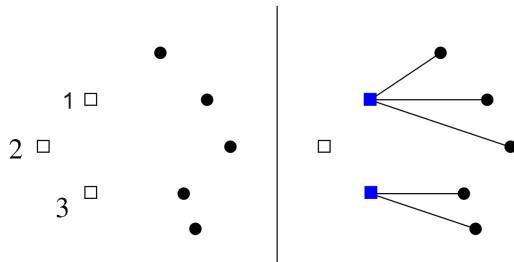
## PLI: Exemplo 2 (Facility Location)

Uma empresa conhece a localização de seus  $n$  clientes denotados por  $j = 1, 2, \dots, n$ . Para atendê-los a companhia decide abrir um conjunto de instalações: cada cliente será atendido por exatamente uma instalação. Uma instalação pode atender vários clientes. As possíveis instalações são denotadas por  $i = 1, 2, \dots, m$ . A empresa conhece o custo de abertura  $f_i$  da instalação  $i$ , e o custo de envio  $c_{ij}$  da instalação  $i$  a um cliente  $j$ .



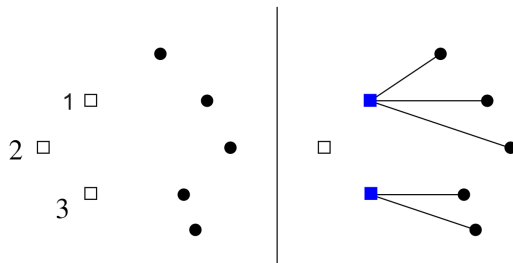
## PLI: Exemplo 2 (Facility Location)

A empresa deve escolher um subconjunto **S** de instalações a serem abertas.  
O objetivo é minimizar o custo total de abertura das instalações mais o custo de envio para cada cliente da instalação que vai atendê-lo.



## PLI: Exemplo 2 (Facility Location)

A empresa deve escolher um subconjunto  $S$  de instalações a serem abertas.  
O objetivo é minimizar o custo total de abertura das instalações mais o custo de envio para cada cliente da instalação que vai atendê-lo.



Note: uma vez escolhido  $S$ , basta associar cada cliente à instalação mais próxima  $\in S$ .



## PLI: Exemplo 2 (Facility Location)

- **Variáveis:**

$y_i = 1$  se abre a instalação  $i$ , e  $y_i = 0$  caso contrário.

$x_{ij} = 1$  se a instalação  $i$  atende  $j$ , e  $x_{ij} = 0$  caso contrário.

## PLI: Exemplo 2 (Facility Location)

- **Variáveis:**

$y_i = 1$  se abre a instalação  $i$ , e  $y_i = 0$  caso contrário.

$x_{ij} = 1$  se a instalação  $i$  atende  $j$ , e  $x_{ij} = 0$  caso contrário.

- **Função objetivo:**  $\sum_{i=1}^m f_i y_i + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$ .

## PLI: Exemplo 2 (Facility Location)

- **Variáveis:**

$y_i = 1$  se abre a instalação  $i$ , e  $y_i = 0$  caso contrário.

$x_{ij} = 1$  se a instalação  $i$  atende  $j$ , e  $x_{ij} = 0$  caso contrário.

- **Função objetivo:**  $\sum_{i=1}^m f_i y_i + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$ .

- **Restrições**

Cada cliente é atendido por exatamente uma instalação:

$$\sum_{i=1}^m x_{ij} = 1 \text{ para } j = 1, \dots, n.$$

## PLI: Exemplo 2 (Facility Location)

- **Variáveis:**

$y_i = 1$  se abre a instalação  $i$ , e  $y_i = 0$  caso contrário.

$x_{ij} = 1$  se a instalação  $i$  atende  $j$ , e  $x_{ij} = 0$  caso contrário.

- **Função objetivo:**  $\sum_{i=1}^m f_i y_i + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$ .

- **Restrições**

Cada cliente é atendido por exatamente uma instalação:

$$\sum_{i=1}^m x_{ij} = 1 \text{ para } j = 1, \dots, n.$$

Cliente só pode ser atendido por uma instalação aberta:

$$x_{ij} \leq y_i \text{ para } i = 1, \dots, m \text{ e } j = 1, \dots, n.$$

## PLI: Exemplo 2 (Facility Location)

- **Variáveis:**

$y_i = 1$  se abre a instalação  $i$ , e  $y_i = 0$  caso contrário.

$x_{ij} = 1$  se a instalação  $i$  atende  $j$ , e  $x_{ij} = 0$  caso contrário.

- **Função objetivo:**  $\sum_{i=1}^m f_i y_i + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$ .

- **Restrições**

Cada cliente é atendido por exatamente uma instalação:

$$\sum_{i=1}^m x_{ij} = 1 \text{ para } j = 1, \dots, n.$$

Cliente só pode ser atendido por uma instalação aberta:

$$x_{ij} \leq y_i \text{ para } i = 1, \dots, m \text{ e } j = 1, \dots, n.$$

- **Não-negatividade:**  $y_i \geq 0$  ( $i = 1, \dots, m$ );  $x_{ij} \geq 0$  ( $i = 1, \dots, m, j = 1, \dots, n$ ).

## PLI: Exemplo 2 (Facility Location)

- **Variáveis:**

$y_i = 1$  se abre a instalação  $i$ , e  $y_i = 0$  caso contrário.

$x_{ij} = 1$  se a instalação  $i$  atende  $j$ , e  $x_{ij} = 0$  caso contrário.

- **Função objetivo:**  $\sum_{i=1}^m f_i y_i + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$ .

- **Restrições**

Cada cliente é atendido por exatamente uma instalação:

$$\sum_{i=1}^m x_{ij} = 1 \text{ para } j = 1, \dots, n.$$

Cliente só pode ser atendido por uma instalação aberta:

$$x_{ij} \leq y_i \text{ para } i = 1, \dots, m \text{ e } j = 1, \dots, n.$$

- **Não-negatividade:**  $y_i \geq 0$  ( $i = 1, \dots, m$ );  $x_{ij} \geq 0$  ( $i = 1, \dots, m, j = 1, \dots, n$ ).

- **Integralidade:**  $y_i \in \{0, 1\}$  ( $i = 1, \dots, m$ );  $x_{ij} \in \{0, 1\}$  ( $i = 1, \dots, m, j = 1, \dots, n$ ).

## PLI: Exemplo 3 (Emparelhamento Máximo)

Um **emparelhamento** em um grafo (não direcionado)  $G = (V, E)$  é um conjunto de arestas que não possuem vértices em comum.

## PLI: Exemplo 3 (Emparelhamento Máximo)

Um **emparelhamento** em um grafo (não direcionado)  $G = (V, E)$  é um conjunto de arestas que não possuem vértices em comum.

Em outras palavras, cada vértice do grafo precisa aparecer em, no máximo, uma aresta do emparelhamento.



## PLI: Exemplo 3 (Emparelhamento Máximo)

Um **emparelhamento** em um grafo (não direcionado)  $G = (V, E)$  é um conjunto de arestas que não possuem vértices em comum.

Em outras palavras, cada vértice do grafo precisa aparecer em, no máximo, uma aresta do emparelhamento.

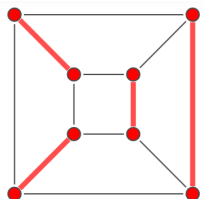
Um **emparelhamento máximo** (ou emparelhamento de cardinalidade máxima) é um emparelhamento que possui a maior quantidade possível de arestas.

**Aplicações:** química orgânica, outorgamento de vagas em universidades, leilões ...

# PLI: Exemplo 3 (Emparelhamento Máximo)

## Problema do Emparelhamento Máximo

Dado um grafo  $G = (V, E)$ , encontrar emparelhamento  $M \subseteq E$  de tamanho máximo.



## PLI: Exemplo 3 (Emparelhamento Máximo)

- **Variáveis:** Para cada aresta  $e$  do grafo, defina uma variável  $x_e$ :

$$x_e = \begin{cases} 1 & \text{se a aresta } e \text{ está no emparelhamento,} \\ 0 & \text{se a aresta } e \text{ não está no emparelhamento.} \end{cases}$$

## PLI: Exemplo 3 (Emparelhamento Máximo)

- **Variáveis:** Para cada aresta  $e$  do grafo, defina uma variável  $x_e$ :

$$x_e = \begin{cases} 1 & \text{se a aresta } e \text{ está no emparelhamento,} \\ 0 & \text{se a aresta } e \text{ não está no emparelhamento.} \end{cases}$$

- **Função objetivo:**  $\sum_{e \in E} x_e$ .

## PLI: Exemplo 3 (Emparelhamento Máximo)

- **Variáveis:** Para cada aresta  $e$  do grafo, defina uma variável  $x_e$ :

$$x_e = \begin{cases} 1 & \text{se a aresta } e \text{ está no emparelhamento,} \\ 0 & \text{se a aresta } e \text{ não está no emparelhamento.} \end{cases}$$

- **Função objetivo:**  $\sum_{e \in E} x_e$ .

- **Restrições:**

Cada vértice pode ter no máximo uma aresta do emparelhamento incidente a ele:

$$\sum_{e \in \delta(v)} x_e \leq 1, \text{ para cada } v \in V.$$

## PLI: Exemplo 3 (Emparelhamento Máximo)

- **Variáveis:** Para cada aresta  $e$  do grafo, defina uma variável  $x_e$ :

$$x_e = \begin{cases} 1 & \text{se a aresta } e \text{ está no emparelhamento,} \\ 0 & \text{se a aresta } e \text{ não está no emparelhamento.} \end{cases}$$

- **Função objetivo:**  $\sum_{e \in E} x_e$ .
- **Restrições:**  
Cada vértice pode ter no máximo uma aresta do emparelhamento incidente a ele:  
 $\sum_{e \in \delta(v)} x_e \leq 1$ , para cada  $v \in V$ .
- **Não-negatividade:**  $x_e \geq 0$ , para cada  $e \in E$ .

## PLI: Exemplo 3 (Emparelhamento Máximo)

- **Variáveis:** Para cada aresta  $e$  do grafo, defina uma variável  $x_e$ :

$$x_e = \begin{cases} 1 & \text{se a aresta } e \text{ está no emparelhamento,} \\ 0 & \text{se a aresta } e \text{ não está no emparelhamento.} \end{cases}$$

- **Função objetivo:**  $\sum_{e \in E} x_e$ .
- **Restrições:**  
Cada vértice pode ter no máximo uma aresta do emparelhamento incidente a ele:  
 $\sum_{e \in \delta(v)} x_e \leq 1$ , para cada  $v \in V$ .
- **Não-negatividade:**  $x_e \geq 0$ , para cada  $e \in E$ .
- **Integralidade:**  $x_e \in \{0, 1\}$ , para cada  $e \in E$ .

## PLI: Exemplo 3 (Emparelhamento Máximo)

**Formulação de PLI para o Problema do Emparelhamento Máximo:**

$$\begin{aligned} & \max \sum_{e \in E} x_e \\ & \text{sujeito a } \sum_{e \in \delta(v)} x_e \leq 1 \text{ para cada } v \in V \\ & x_e \in \{0, 1\} \text{ para cada } e \in E \end{aligned}$$



# PLI: Exemplo 3 (Emparelhamento Máximo)

**Formulação de PLI para o Problema do Emparelhamento Máximo:**

$$\begin{aligned} & \max \sum_{e \in E} x_e \\ & \text{sujeito a } \sum_{e \in \delta(v)} x_e \leq 1 \text{ para cada } v \in V \\ & \quad x_e \in \{0, 1\} \text{ para cada } e \in E \end{aligned}$$

A formulação possui  $|E| = m$  variáveis e  $|V| = n$  restrições.

## PLI: Exemplo 3 (Emparelhamento Máximo)

- Um geral, para qualquer grafo  $G = (V, E)$ , encontrar um emparelhamento máximo é um problema difícil: não há algoritmo eficiente conhecido.

## PLI: Exemplo 3 (Emparelhamento Máximo)

- Um geral, para qualquer grafo  $G = (V, E)$ , encontrar um emparelhamento máximo é um problema difícil: não há algoritmo eficiente conhecido.
- **E se o grafo  $G$  for bipartido?** - Existe um algoritmo **não trivial**, porém eficiente para este caso particular.

## PLI: Exemplo 3 (Emparelhamento Máximo)

- Um geral, para qualquer grafo  $G = (V, E)$ , encontrar um emparelhamento máximo é um problema difícil: não há algoritmo eficiente conhecido.
- **E se o grafo  $G$  for bipartido?** - Existe um algoritmo **não trivial**, porém eficiente para este caso particular.
- Não temos algoritmo eficiente para resolver a nossa formulação de **PLI**, mas temos algoritmo eficiente para resolver **PL**.

## PLI: Exemplo 3 (Emparelhamento Máximo)

- Um geral, para qualquer grafo  $G = (V, E)$ , encontrar um emparelhamento máximo é um problema difícil: não há algoritmo eficiente conhecido.
- **E se o grafo  $G$  for bipartido?** - Existe um algoritmo **não trivial**, porém eficiente para este caso particular.
- Não temos algoritmo eficiente para resolver a nossa formulação de **PLI**, mas temos algoritmo eficiente para resolver **PL**.
- **Ideia:** vamos “esquecer” (*relaxar*) as restrições de **integralidade** das variáveis (supomos que  $0 \leq x_e \leq 1$ ). Vamos resolver este problema de **PL** “relaxado”.

## PLI: Exemplo 3 (Emparelhamento Máximo)

- É possível mostrar que, se  $G$  é **bipartido**, sempre podemos encontrar uma solução dessa formulação do problema de **PL** relaxado com variáveis inteiras.

## PLI: Exemplo 3 (Emparelhamento Máximo)

- É possível mostrar que, se  $G$  é **bipartido**, sempre podemos encontrar uma solução dessa formulação do problema de **PL** relaxado com variáveis inteiras.
- **Essa solução inteira é necessariamente ótima, encontrada em tempo polinomial!**

## PLI: Exemplo 3 (Emparelhamento Máximo)

- É possível mostrar que, se  $G$  é **bipartido**, sempre podemos encontrar uma solução dessa formulação do problema de **PL** relaxado com variáveis inteiras.
- **Essa solução inteira é necessariamente ótima, encontrada em tempo polinomial!**
- Isso mostra que, mesmo não havendo um algoritmo geral eficiente para **PLI**, podemos usar as formulações de **PLI** para resolver diversos subproblemas e instâncias específicas de forma eficiente.



## PLI: Exemplo 3 (Emparelhamento Máximo)

- É possível mostrar que, se  $G$  é **bipartido**, sempre podemos encontrar uma solução dessa formulação do problema de **PL** relaxado com variáveis inteiras.
- **Essa solução inteira é necessariamente ótima, encontrada em tempo polinomial!**
- Isso mostra que, mesmo não havendo um algoritmo geral eficiente para **PLI**, podemos usar as formulações de **PLI** para resolver diversos subproblemas e instâncias específicas de forma eficiente.
- Ou seja, usar **PLI** é interessante para problemas difíceis, inclusive ele pode “detectar” casos nos quais é possível resolver o problema de forma eficiente.

# Programação Linear – Importância

- **Lembrando:** Uma formulação como **PL** de um problema **P** qualquer, nada mais é do que uma **redução** de  $P \propto PL$ . A redução ( $P \propto_{poli} PL$  ou  $P \propto_{poli} PLI$ ) é polinomial somente se o número de restrições e variáveis é polinomial no tamanho da instância de **P**.

# Programação Linear – Importância

- **Lembrando:** Uma formulação como **PL** de um problema **P** qualquer, nada mais é do que uma **redução** de  $P \propto PL$ . A redução ( $P \propto_{poli} PL$  ou  $P \propto_{poli} PLI$ ) é polinomial somente se o número de restrições e variáveis é polinomial no tamanho da instância de **P**.
- Suponha que **P** pode ser formulado como um **PL**. Como **PL** pode ser resolvido em tempo polinomial através de um algoritmo de pontos interiores, se a formulação de **PL** de **P** for polinomial, então temos um **algoritmo polinomial para P**.

# Programação Linear – Importância

- **Lembrando:** Uma formulação como **PL** de um problema **P** qualquer, nada mais é do que uma **redução** de  $P \propto PL$ . A redução ( $P \propto_{poli} PL$  ou  $P \propto_{poli} PLI$ ) é polinomial somente se o número de restrições e variáveis é polinomial no tamanho da instância de **P**.
- Suponha que **P** pode ser formulado como um **PL**. Como **PL** pode ser resolvido em tempo polinomial através de um algoritmo de pontos interiores, se a formulação de **PL** de **P** for polinomial, então temos um **algoritmo polinomial para P**.

**É importante saber formular modelos de PL.**

# Programação Linear Inteira – Importância

- Suponha que não conhecemos um algoritmo eficiente para **P** (acreditamos que não há algoritmo eficiente), e **P** pode ser formulado como um **PLI**.

# Programação Linear Inteira – Importância

- Suponha que não conhecemos um algoritmo eficiente para **P** (acreditamos que não há algoritmo eficiente), e **P** pode ser formulado como um **PLI**.
- Se a formulação de **PLI** for polinomial, então temos uma grande quantidade de cientistas e pesquisadores que criaram ferramentas poderosas para resolver a formulação de **P** da maneira mais rápida possível, ou resolver subproblemas e instâncias específicas de **P** de forma eficiente.

# Programação Linear Inteira – Importância

- Suponha que não conhecemos um algoritmo eficiente para **P** (acreditamos que não há algoritmo eficiente), e **P** pode ser formulado como um **PLI**.
- Se a formulação de **PLI** for polinomial, então temos uma grande quantidade de cientistas e pesquisadores que criaram ferramentas poderosas para resolver a formulação de **P** da maneira mais rápida possível, ou resolver subproblemas e instâncias específicas de **P** de forma eficiente.

**É importante saber formular modelos de PLI.**

# Software para PL e PLI

Implementar um algoritmo para resolver PLs é **altamente não trivial**. Problemas reais na indústria podem ter milhões de variáveis. Geralmente usamos um software existente.



# Software para PL e PLI

Implementar um algoritmo para resolver PLs é **altamente não trivial**. Problemas reais na indústria podem ter milhões de variáveis. Geralmente usamos um software existente.

## Programas comerciais:

- CPLEX / IBM: <https://www.ibm.com/products/ilog-cplex-optimization-studio>
- XPRESS: <https://www.fico.com/en/products/fico-xpress-optimization>
- GUROBI: <https://www.gurobi.com/>

# Software para PL e PLI

Implementar um algoritmo para resolver PLs é **altamente não trivial**. Problemas reais na indústria podem ter milhões de variáveis. Geralmente usamos um software existente.

## Programas comerciais:

- CPLEX / IBM: <https://www.ibm.com/products/ilog-cplex-optimization-studio>
- XPRESS: <https://www.fico.com/en/products/fico-xpress-optimization>
- GUROBI: <https://www.gurobi.com/>

## Programas livres:

- SCIP: <https://www.scipopt.org/>
- GLPK / Gnu: <https://www.gnu.org/software/glpk/glpk.html>
- COIN-OR: <https://www.coin-or.org/Clp> | <https://www.coin-or.org/downloading>

# Resumo

- 1 Revisão do conteúdo e objetivo
- 2 Programação Linear
- 3 **Síntese**

# Síntese

- Um problema de **PL** ou **PLI** pode não ter nenhuma **solução viável** – ser **inviável**. Isso significa que não há nenhuma solução para o meu problema.
- Um problema de **PL** ou **PLI** que é **viável** pode não ter nenhuma **solução ótima** melhor do que todas as outras soluções viáveis, sendo **ilimitado**.
- Um problema de **PL** ou **PLI** pode ter uma ou várias **soluções ótimas**.
- Usar **PLI** é interessante para problemas difíceis: cientistas e pesquisadores criaram essa ferramenta para resolver formulações, subproblemas e instâncias específicas da maneira mais rápida possível.



## Dúvidas

# Dúvidas?