Projeto e Análise de Algoritmos II (MC558)
Introdução à Programação Linear

Prof. Dr. Ruben Interian

Resumo

Resumo

Revisão do conteúdo

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

Objetivo

Responder as perguntas:

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

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

Resumo

PL vs PLI

Programação Linear (**PL**):

Entrada: Matriz $A_{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.
Objetivo: Achar o vetor $x \in \mathbb{R}^n$ que minimize $c^T x$, sujeito a $Ax \leq b, x \geq 0$.

$$\min c^T x$$
$$\text{sujeito a } Ax \leq b, x \geq 0$$

Programação Linear Inteira (**PLI**):

Entrada: Matriz $A_{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.
Objetivo: Achar o vetor $x \in \mathbb{Z}^n$ que minimize $c^T x$, sujeito a $Ax \leq b, x \geq 0$.

$$\min c^T x$$
$$\text{sujeito a } Ax \leq b, x \geq 0$$

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**?
- Se existe um algoritmo para resolver **PL** / **PLI**, que tipo de saída eles oferecem?

PL e PLI – soluções viáveis e soluções ótimas

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

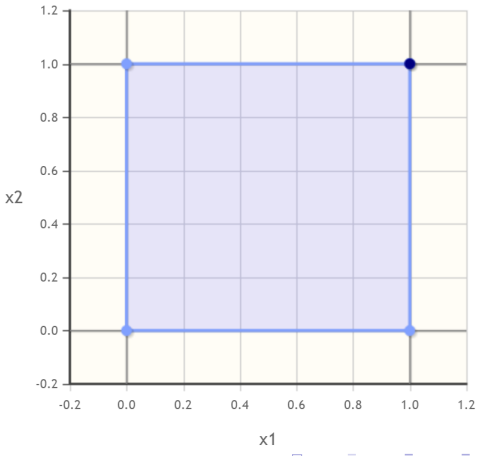
$$\begin{aligned} \max\ & x_1 + x_2 \\ \text{sujeito a}\ & x_1 \leq 1 \\ & x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

PL e PLI – soluções viáveis e soluções ótimas

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

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

No caso da **PL**, sempre **possui 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**.

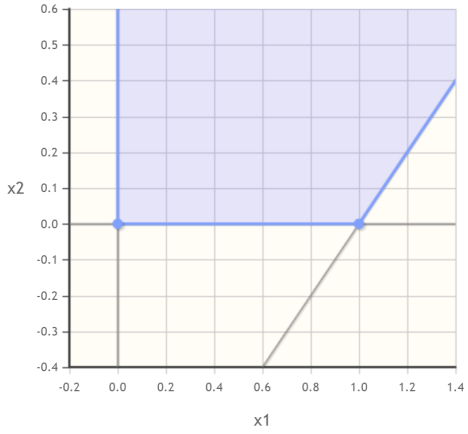
$$\begin{aligned} \max\ & x_1 + x_2 \\ \text{sujeito a } & x_1 - x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

PL e PLI – soluções viáveis e soluções ótimas

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

$$\max\ x_1 + x_2$$
$$\text{sujeito a } x_1 - x_2 \leq 1$$
$$x_1, x_2 \geq 0$$

Podemos não ter ótimo ...



PL e PLI – soluções viáveis e soluções ótimas

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

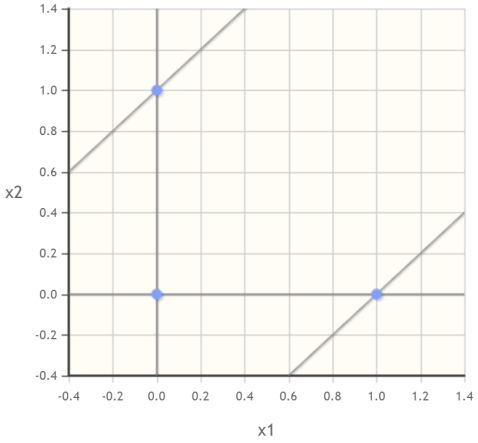
$$\begin{aligned} \max\ & x_1 + x_2 \\ \text{sujeito a}\ & x_1 - x_2 \geq 1 \\ & x_1 - x_2 \leq -1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

PL e PLI – soluções viáveis e soluções ótimas

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

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

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**
- Se há alguma **solução viável**, sempre existe uma **solução ótima**? – **Não**

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 $i = 1, \ldots, n$. Queremos encontrar um conjunto de itens com o maior valor total possível, que não ultrapasse a capacidade da mochila.

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 $i = 1, \ldots, n$. 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 $i = 1, \ldots, n$, defina uma variável x_i:

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

PLI: Exemplo 1 (Problema da Mochila)

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

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

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

PLI: Exemplo 1 (Problema da Mochila)

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

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

- **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 $i = 1, \ldots, n$, defina uma variável x_i:

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

- **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 $i = 1, \ldots, n$.

PLI: Exemplo 1 (Problema da Mochila)

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

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

- **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 $i = 1, \ldots, n$.
- **Integralidade**: $x_i \in \{0, 1\}$, para cada $i = 1, \ldots, n$.

PLI: Exemplo 1 (Problema da Mochila)

Formulação de PLI para o Problema da Mochila:

$$\max \sum_{i=1}^{n} v_i x_i$$

$$\text{sujeito a } \sum_{i=1}^{n} w_i x_i \leq W$$

$$x_i \in \{0, 1\} \text{ para cada } i = 1, \ldots, n$$

PLI: Exemplo 1 (Problema da Mochila)

Formulação de PLI para o Problema da Mochila:

$$\max \sum_{i=1}^{n} v_i x_i$$

$$\text{sujeito a } \sum_{i=1}^{n} w_i x_i \leq W$$

$$x_i \in \{0,1\} \text{ para cada } i = 1, \ldots, n$$

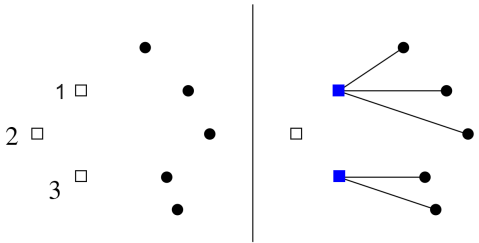
A formulação possui n variáveis e apenas 1 restrição!

PLI: Exemplo 2 (Facility Location)

Uma empresa conhece a localização de seus n clientes denotados por $j = 1, 2, \ldots, n$. 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 $i = 1, 2, \ldots, m$. 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.

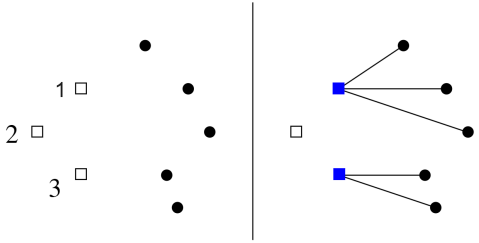
PLI: Exemplo 2 (Facility Location)

Uma empresa conhece a localização de seus n clientes denotados por $j = 1, 2, \ldots, n$. 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 $i = 1, 2, \ldots, m$. 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.



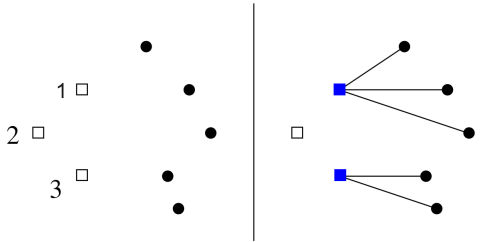
PLI: Exemplo 2 (Facility Location)

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 que uma vez escolhido S, basta associar cada cliente à instalação aberta mais próxima.

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

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:
  $\sum_{i=1}^{m} x_{ij} = 1$ para $j = 1, \ldots, n$.
  Cliente só pode ser atendido por uma instalação aberta:
  $x_{ij} \leq y_i$ para $i = 1, \ldots, m$ e $j = 1, \ldots, n$.

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:
  $\sum_{i=1}^{m} x_{ij} = 1$ para $j = 1, \ldots, n$.
  Cliente só pode ser atendido por uma instalação aberta:
  $x_{ij} \leq y_i$ para $i = 1, \ldots, m$ e $j = 1, \ldots, n$.
- **Não-negatividade**: $y_i \geq 0$ $(i = 1, \ldots, m)$; $x_{ij} \geq 0$ $(i = 1, \ldots, m, j = 1, \ldots, n)$.

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:
  $\sum_{i=1}^{m} x_{ij} = 1$ para $j = 1, \ldots, n$.
  Cliente só pode ser atendido por uma instalação aberta:
  $x_{ij} \leq y_i$ para $i = 1, \ldots, m$ e $j = 1, \ldots, n$.
- **Não-negatividade**: $y_i \geq 0$ $(i = 1, \ldots, m)$; $x_{ij} \geq 0$ $(i = 1, \ldots, m,\ j = 1, \ldots, n)$.
- **Integralidade**: $y_i \in \{0, 1\}$ $(i = 1, \ldots, m)$; $x_{ij} \in \{0, 1\}$ $(i = 1, \ldots, m,\ j = 1, .., n)$.

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.

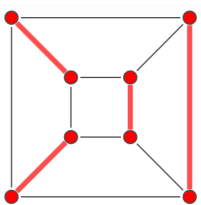
Um **emparelhamento máximo** (ou emparelhamento de cardinalidade máxima) é um emparelhamento que possui a maior quantidade possível de arestas.

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

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:

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

PLI: Exemplo 3 (Emparelhamento Máximo)

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

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

- **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:

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

- **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 $v \in V$.

PLI: Exemplo 3 (Emparelhamento Máximo)

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

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

- **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:

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

- **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:

$$\max \sum_{e \in E} x_e$$

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

$$x_e \in \{0, 1\} \text{ para cada } e \in E$$

PLI: Exemplo 3 (Emparelhamento Máximo)

Formulação de PLI para o Problema do Emparelhamento Máximo:

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

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 sempre que o número de restrições e variáveis seja 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 sempre que o número de restrições e variáveis seja 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 sempre que o número de restrições e variáveis seja 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

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.

Material bibliográfico e exercícios

U. Manber. Introduction to Algorithms. – **Cap. 10**.

Exercícios: ver exercícios no final do Capítulo 10.

T. Cormen et al. Algoritmos - Teoria e Prática (3a ed.). – **Cap. 29**.

Dúvidas

Dúvidas?