

Caminho Mínimo: Uma Modelagem com Programação Linear

Claudio Cesar de Sá¹

Pesquisador Independente

Roteiro

1. Complexidade de Problemas
2. O que é a Programação Linear Inteira (PLI)
3. Um problema: Caminho Mínimo
4. Modelagem com uma técnica de PLI ⇒ **este vídeo**
5. Implementação e código com OR-TOOLS (Python) ⇒
próximo vídeo
6. Generalizando o Caminho Mínimo (avançado) ⇒ **um outro vídeo**
7. Este material: https://github.com/claudiosa/CCS/tree/master/presentations-seminars/cam_min_PL

Classes de Problemas

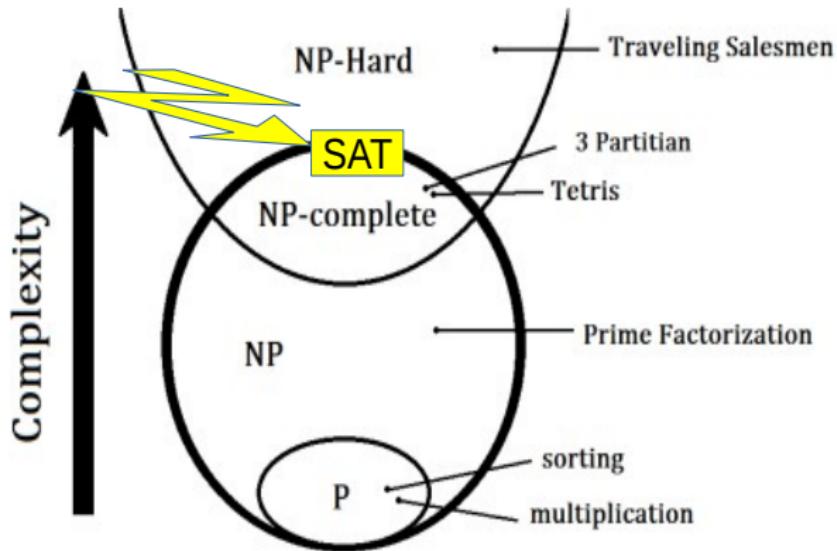
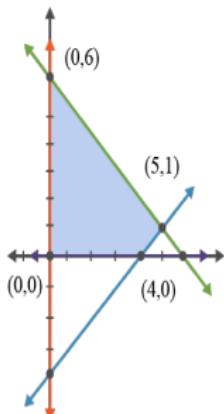


Figura: Alguns problemas e suas complexidades

Programação Linear

constraints:
$$\begin{cases} x + y \geq 6 \\ x - y \geq 4 \\ x \geq 0 \\ y \geq 0 \end{cases}$$

Objective Function: $C = 2x + y$



$$(0,0): C = 2(0) + (0) = 0 \text{ minimum}$$

$$(0,6): C = 2(0) + (6) = 6$$

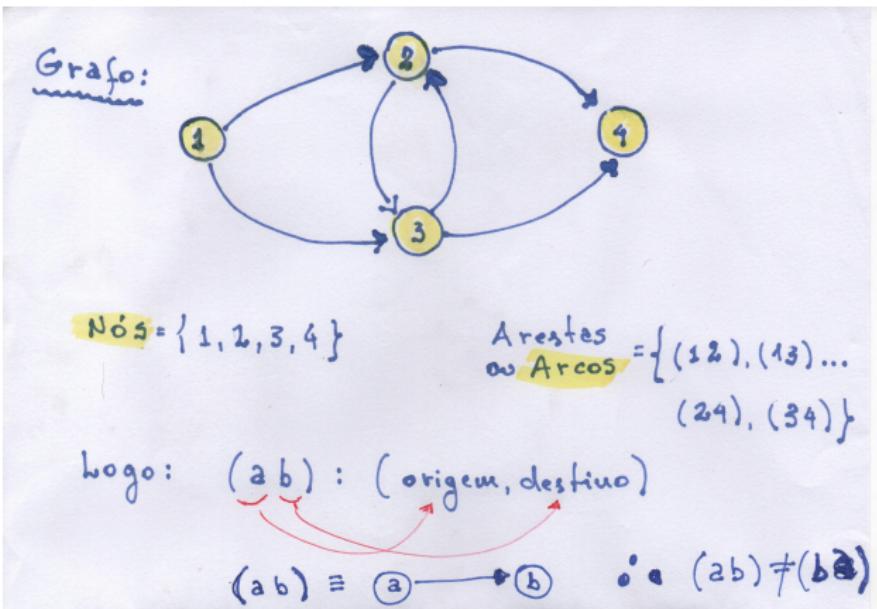
$$(5,1): C = 2(5) + (1) = 11 \text{ maximum}$$

$$(4,0): C = 2(4) + (0) = 8$$

Calcworkshop.com

Figura: Uma técnica de resolver problemas: equações lineares

Elementos de um Grafo



Problema do Caminho Mínimo

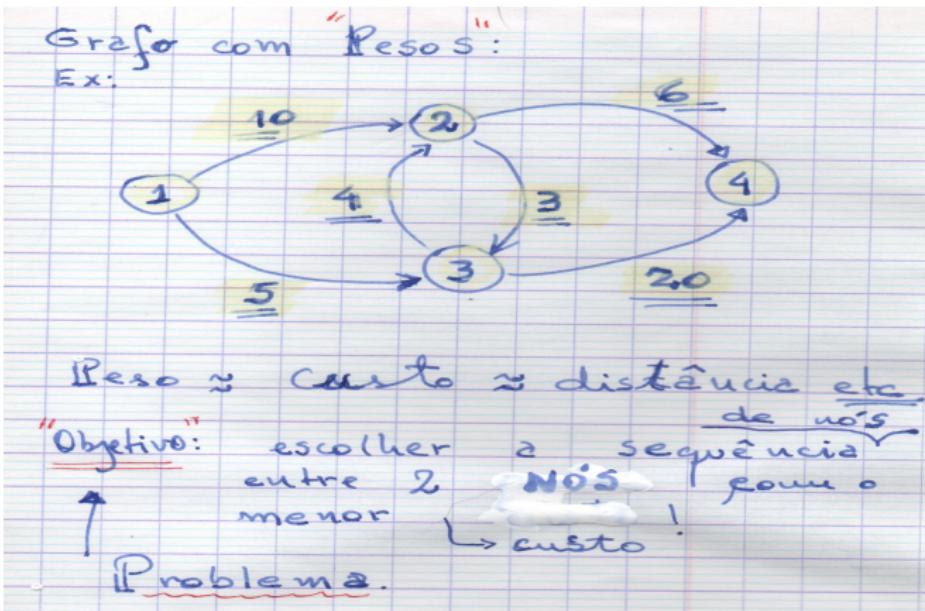


Figura: Um problema da classe P (mas muito útil) com uma solução via PLI

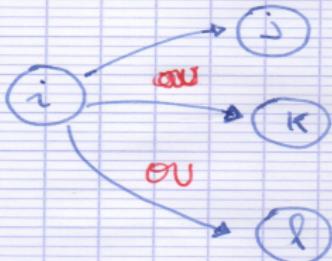
Referência:

https://en.wikipedia.org/wiki/Shortest_path_problem

Variável de Decisão e Uso

VARIÁVEL DE DECISÃO: ("decide algo")

Ex: $X = \{0, 1\}$; $\{\text{F}, \text{V}\}$, ...
(binária ésta!)



Qual caminho
é tomar?

$$x_{ij} = 1/0$$

$$\underline{x_{ik}} = 1/0$$

$$\underline{x_{il}} = 1/0$$

Só pode 1 arco ser escolhido!

Logo: $x_{ij} + x_{ik} + x_{il} = 1$

Notação rigorosa:

$$i \rightarrow j \neq j \rightarrow i$$

x origem, destino

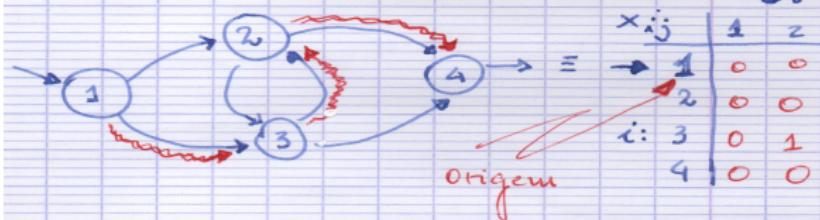
Matriz de Decisão e Uso

X_{ij}	1	2	3	4
1	0/1	0/1	0/1	0/1
2	:	:	:	:
3	:	:	:	:
4	0/1	0/1	0/1	0/1

Ex1:

MATRIZ OU VARIÁVEL DE DECISÃO

$\begin{matrix} \text{3:} \\ \text{Destino} \end{matrix}$



x_{ij}	1	2	3	4
1	0	0	1	0
2	0	0	0	1
3	0	1	0	0
4	0	0	0	0

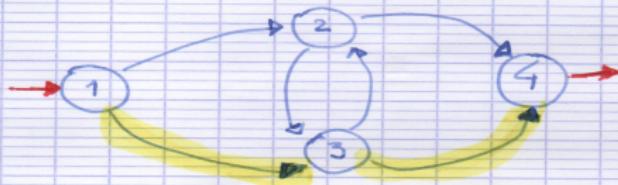
Escolhidos:
(SEQUÊNCIA)



$$\text{Logo } x_{13} = x_{32} = x_{24} = 1$$

Definindo um Caminho

Logo um CAMINHO é uma SEQUÊNCIA
(DE NÓS escolhidos):



Ex:

$$2:4 \Rightarrow 1 \rightarrow 2 \rightarrow 4$$

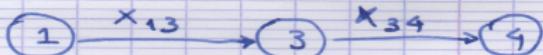
$$1 \rightarrow 3 \rightarrow 4$$

$$\text{Ex}_1 \Rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \quad (\text{exemplo anterior})$$

...
...

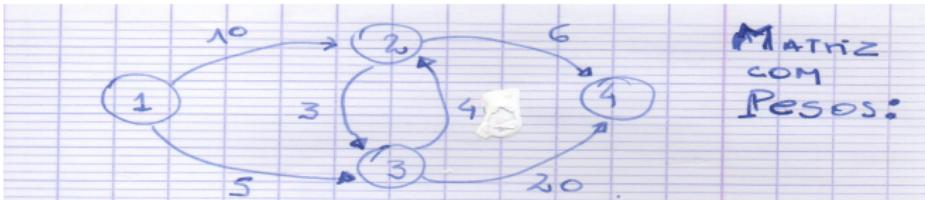
Se repetir nó \Rightarrow Ineficiente!

Ex₂:



$$\therefore x_{13} = x_{34} = 1 \quad \leftarrow \text{os demais iguais a zero}$$

O Caminho e a Matriz de Peso



juntando: caminho + MATRIZ DECISÃO
para um caminho x_{ij}

$$1 \rightarrow 4$$

A solução seria: $1 \rightarrow 3, 3 \rightarrow 2, 2 \rightarrow 4$

x_{ij}	1	2	3	4
1	0	0	1	0
2	0	0	0	1
3	0	1	0	0
4	0	0	0	0

Como
Selecionar
 x_{13}, x_{32} e
 x_{24} ?

{ Mas como encontrar esta combinação ideal? }
{ Menor CAMINHO = MENOR CUSTO }

Formalizando os Elementos

d_{ij}	$j:$			
$i:$	1	2	3	4
1	Ø	10	5	M
2	M	Ø	3	6
3	M	4	Ø	20
4	M	M	M	Ø

Matriz - peso dos arcos
(Peso ou valor...)

Matriz ponderada

$i:$ origem
 $j:$ destino

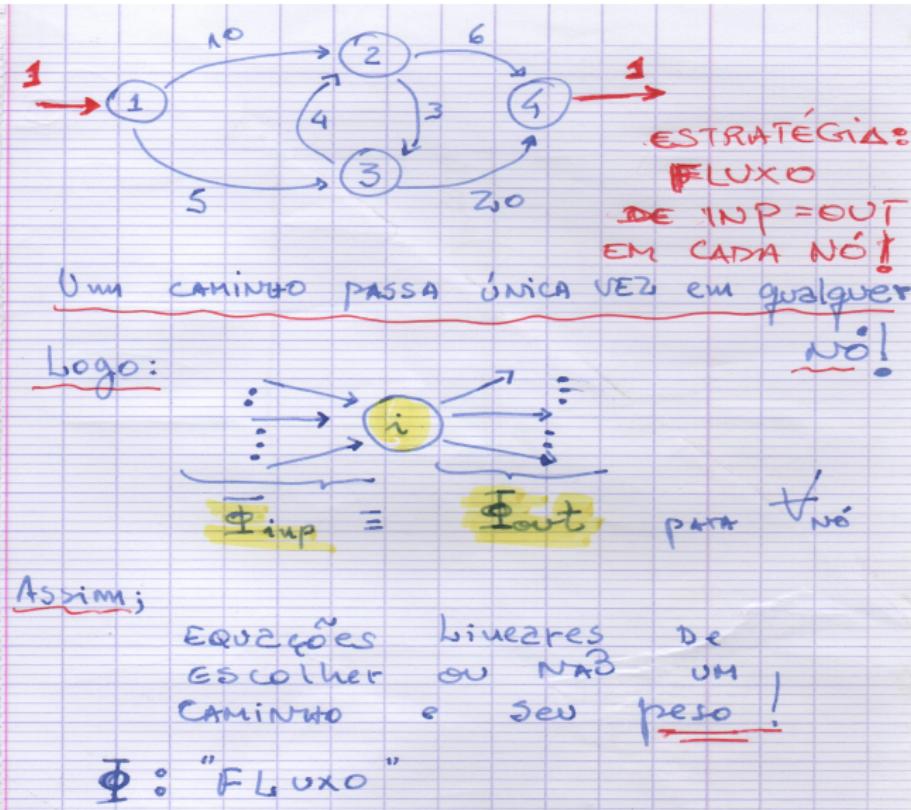
d : distância, custo, capacidade, ... etc

M : um valor negativo ou "grande" (seu conexão)
Ex: $M = 9999$

BASICAMENTE UM PRODUTO ESCALAR

$\approx \text{Peso}_{ij} \times x_{ij}$ e minimizar estas escolhas !

Estratégia deste Problema: Fluxo (Φ)



Equações Finais

$$\Phi_{iup}^i (\text{fluxo}) = \Xi_{out}^i (\text{fluxo})$$

$$\#1: \underline{\underline{1}} = \underline{x_{13}} + \underline{x_{12}}$$

$$\#2: \underline{x_{12}} + \underline{x_{32}} = \underline{x_{23}} + \underline{x_{24}}$$

$$\#3: \underline{x_{13}} + \underline{x_{23}} = \underline{x_{32}} + \underline{x_{34}}$$

$$\#4: \underline{x_{24}} + \underline{x_{34}} = \underline{\underline{1}}$$

PARA escolha do caminho mínimo

é minimizar $= \sum_{(ij) \in C} d_{ij} x_{ij}$

i.e:

Para o caso acima é minimizar estás arestas ponderadas:

$$\begin{aligned} f_{\text{mín}} = & 5x_{13} + 40x_{12} + \\ & 3x_{23} + 6x_{24} + \\ & 4x_{32} + 20x_{34} \end{aligned}$$

arestas
//
//
//
TODAS!

Próximos Passos:

- ▶ Implementar na OR-TOOLS com Python (próximo vídeo)
- ▶ Ferramenta livre mantida pela Google
- ▶ Suporta várias linguagens de *front-end*: C++, C#, Java e Python
- ▶ Vários *solvers*
- ▶ Vamos usar o *solver*: CP-SAT
- ▶ CP: *Constraint Programming*
- ▶ *Obrigado!*

Contato e Comentários:

- ▶ <https://claudiocesar.wordpress.com/>
- ▶ <https://github.com/claudiosa>
- ▶ Email: claudio.sa@udesc.br
- ▶ Email: ccs1664@gmail.comr
- ▶ *Thank you so much!*