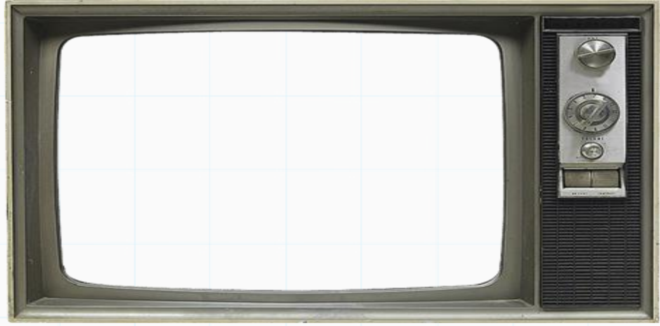
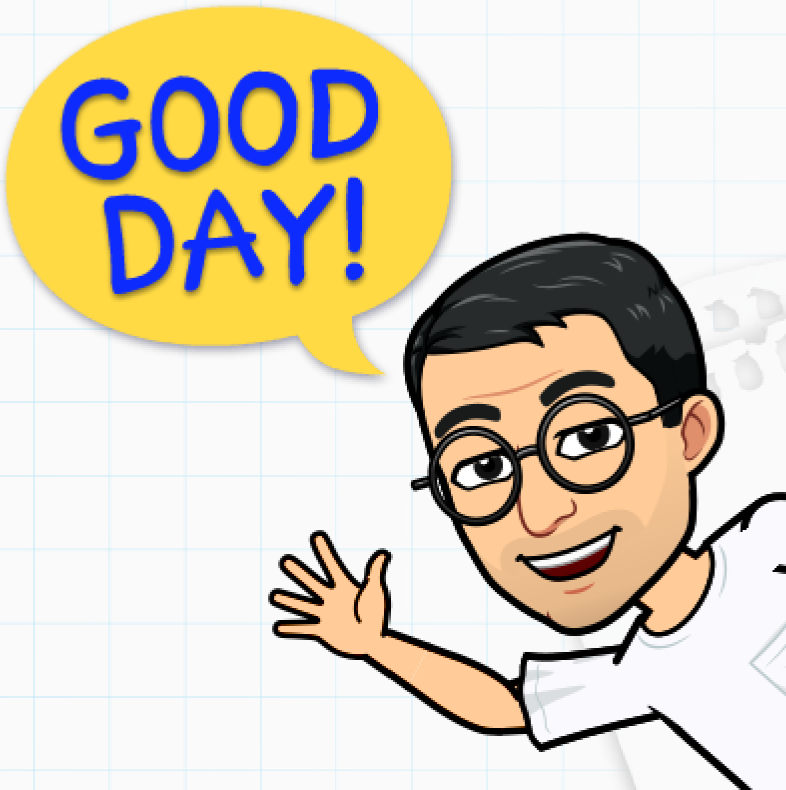
Programação Inteira

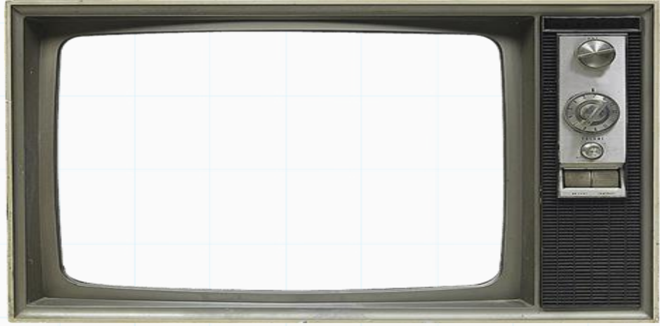


Professor : Yuri Frota

www.ic.uff.br/~yuri/pi.html

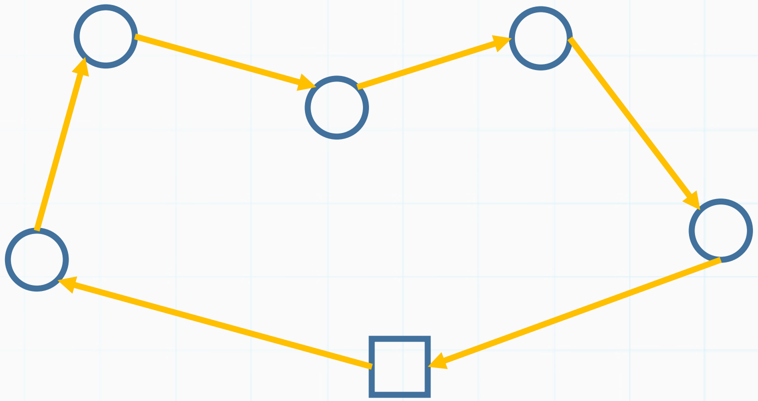
yuri@ic.uff.br





Formulações Clássicas

Problema do Caixeiro Viajante com contratos: Dado um mapa de cidades e um ponto de partida do caixeiro (depósito), temos que encontrar um percurso para o caixeiro visitar (n-1) cidades, voltando ao ponto de partida, estando uma vez em cada cidade, de modo a reduzir o custo financeiro de contrato de aluguel do veículo.

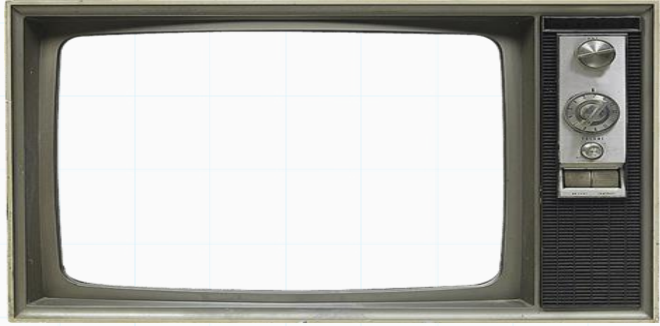


- G=(V,A) direcionada e completo
- c_{ij} custo de viagem do vértice i para j

Existem 3 possibilidades de contrato de veículos usado pelo caixeiro:

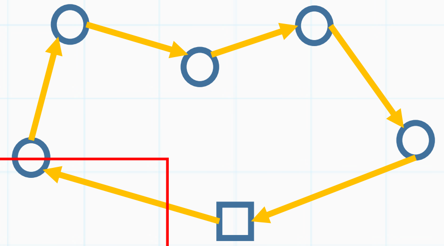
Contrato 1: Existe um custo K_1 por km andando
Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado
Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.

Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

Variáveis:



$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$c_m \in \mathbb{R}^+ : \text{ custo financeiro}$$

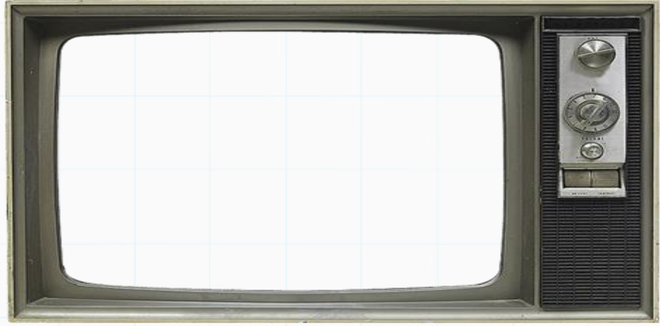
$$d_3 \in \mathbb{R}^+ : \text{ distância percorrida no contrato 3, além de } D_3 \text{ o que vai ser cobrado a mais}$$

Contrato 1: Existe um custo K_1 por km andando

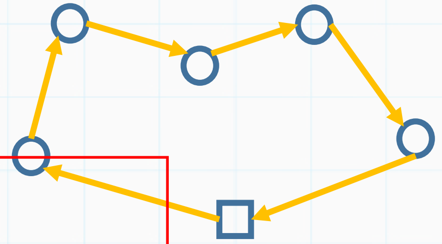
Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.

Formulações Clássicas



Problema do Caixeiro Viajante com contratos:



Variáveis:

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$c_m \in \mathbb{R}^+ : \text{ custo financeiro}$$

$$d_3 \in \mathbb{R}^+ : \text{ distância percorrida no contrato 3, além de } D_3 \text{ o que vai ser cobrado a mais}$$

Contrato 1: Existe um custo K_1 por km andando

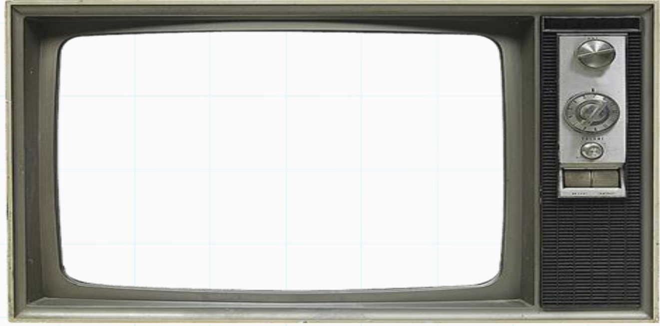
Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.

Função Objetivo:

minimizar custo financeiro

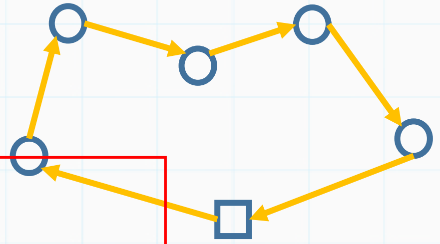
Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

Variáveis:

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$



$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$c_m \in \mathbb{R}^+ : \text{ custo financeiro}$$

$$d_3 \in \mathbb{R}^+ : \text{ distância percorrida no contrato 3, além de } D_3 \text{ o que vai ser cobrado a mais}$$

Contrato 1: Existe um custo K_1 por km andando

Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

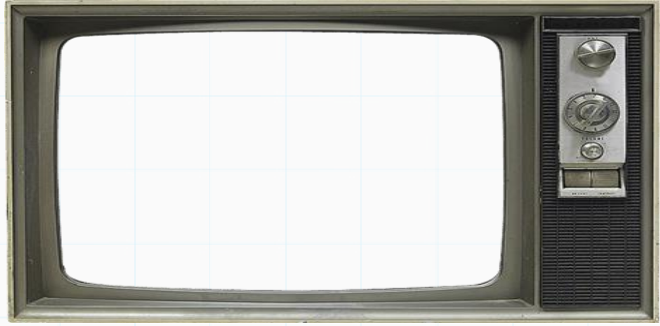
Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.

Função Objetivo:

$$\min \quad c_m$$

minimizar custo financeiro

Formulações Clássicas



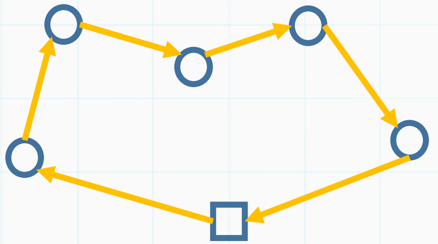
Problema do Caixeiro Viajante com contratos:

Restrições:

$$\sum_{\substack{j \in V \\ i \neq j}} x_{ij} = 1, \quad \forall i \in V$$

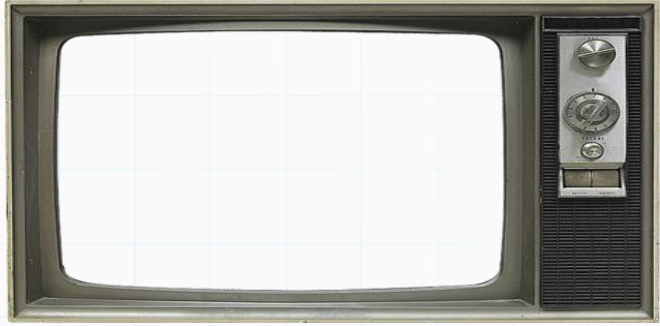
$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1, \quad \forall j \in V$$

todo vértice tem que fazer parte da rota



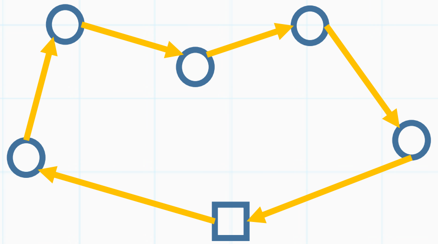
Restrições clássicas do caixeiro

Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

Restrições:



$$\sum_{\substack{j \in V \\ i \neq j}} x_{ij} = 1, \quad \forall i \in V$$

todo vértice tem que fazer parte da rota

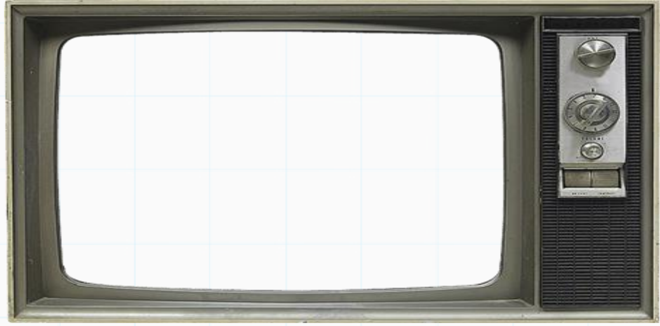
$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1, \quad \forall j \in V$$

Restrições clássicas do caixeiro

$$\sum_{\substack{i,j \in S \\ i \neq j}} x_{ij} \leq |S| - 1, \quad \forall S \subset V, |S| \geq 2$$

subciclo não gerador

Formulações Clássicas

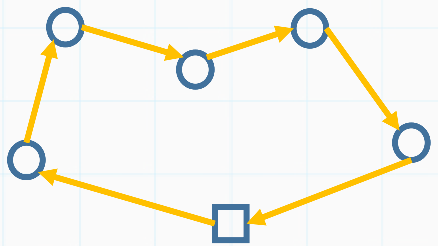


Problema do Caixeiro Viajante com contratos:

Restrições:

$$\sum_{\substack{j \in V \\ i \neq j}} x_{ij} = 1, \quad \forall i \in V$$

todo vértice tem que fazer parte da rota



$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1, \quad \forall j \in V$$

Restrições clássicas do caixeiro

$$\sum_{\substack{i,j \in S \\ i \neq j}} x_{ij} \leq |S| - 1, \quad \forall S \subset V, |S| \geq 2$$

subciclo não gerador

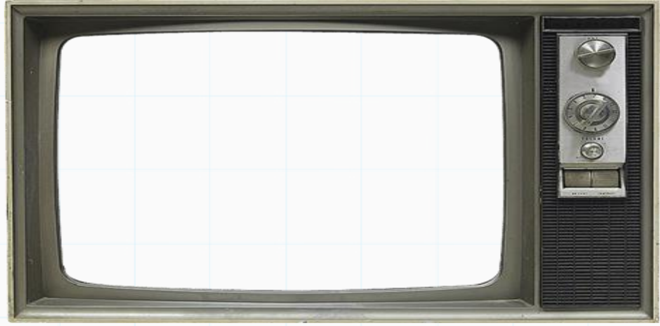
$$x_{ij} \in \{0,1\}, \quad \forall i,j \in V \text{ onde } i \neq j$$

$$c_i \in \{0,1\}, \quad \forall i = \{1,2,3\}$$

$$c_m, d_3 \in \mathbb{R}^+$$

integralidade e ñ negatividade

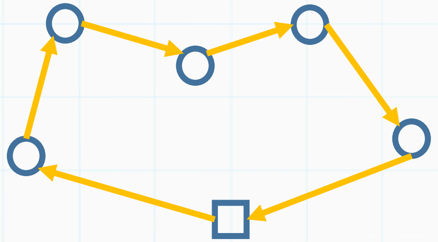
Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

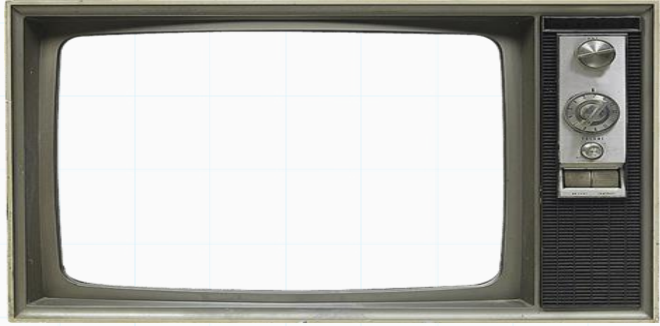
Restrições:

só podemos ter um contrato ativo (c)



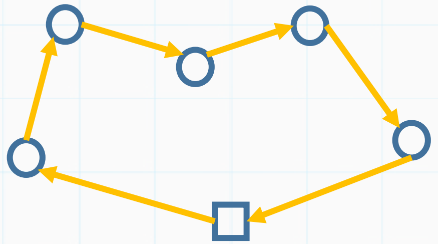
$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

Restrições:

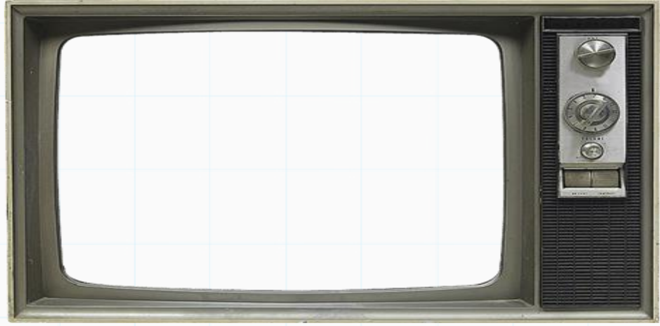


$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)

$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

Formulações Clássicas

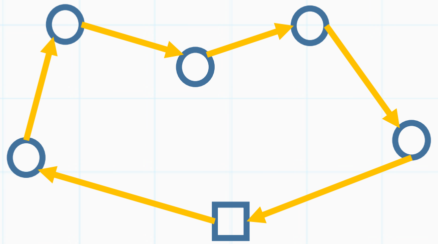


Problema do Caixeiro Viajante com contratos:

Restrições:

$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)



Contrato 1: Existe um custo K_1 por km andando

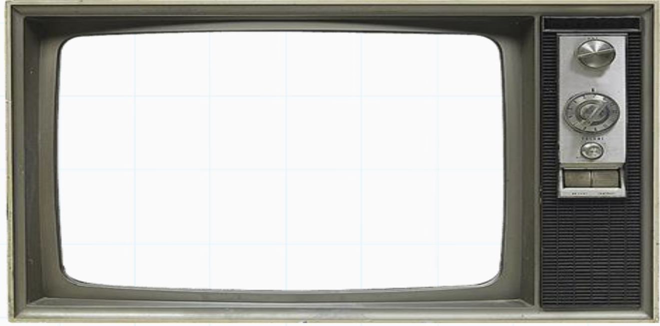
se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

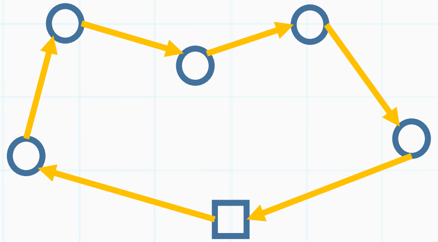
$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1, 2, 3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$c_m \in \mathbb{R}^+ : \text{ custo financeiro}$$

Formulações Clássicas



Problema do Caixeiro Viajante com contratos:



Restrições:

$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)

Contrato 1: Existe um custo K_1 por km andando

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_1) - M(1 - c_1) \leq c_m$$

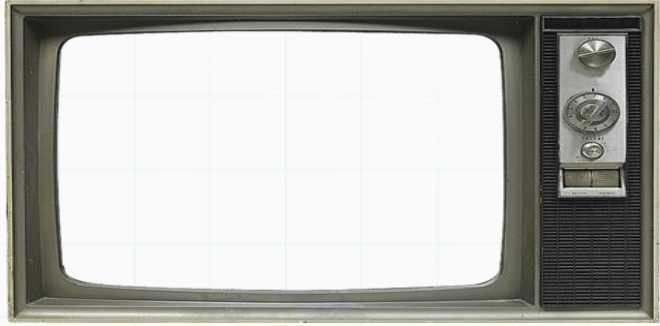
se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

$$c_m \in \mathbb{R}^+ : \text{ custo financeiro}$$

Formulações Clássicas

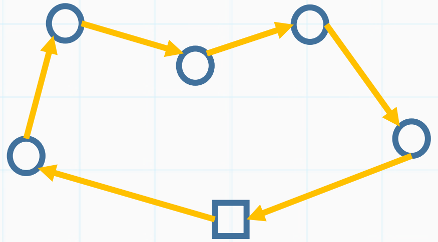


Problema do Caixeiro Viajante com contratos:

Restrições:

$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)



Contrato 1: Existe um custo K_1 por km andando

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_1) - M(1 - c_1) \leq c_m$$

se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

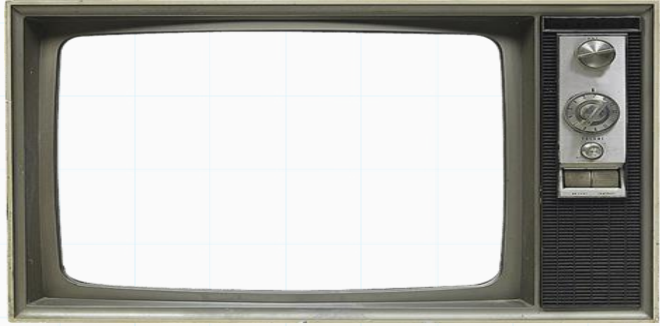
se contrato 2 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

$c_m \in \mathbb{R}^+$: custo financeiro

Formulações Clássicas

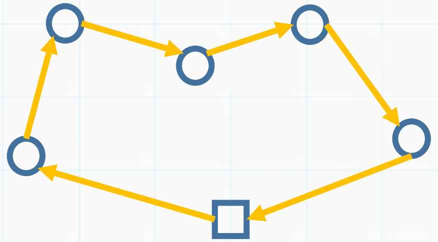


Problema do Caixeiro Viajante com contratos:

Restrições:

$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)



Contrato 1: Existe um custo K_1 por km andando

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_1) - M(1 - c_1) \leq c_m$$

se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_2) + f_2 - M(1 - c_2) \leq c_m$$

se contrato 2 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

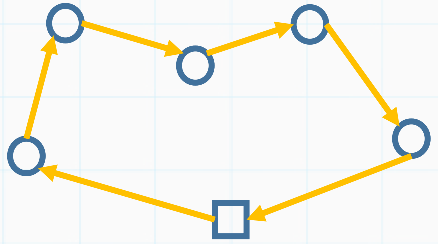
$$c_m \in \mathbb{R}^+ : \text{ custo financeiro}$$

Formulações Clássicas

Problema do Caixeiro Viajante com contratos:

Restrições:

Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.



$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)

Contrato 1: Existe um custo K_1 por km andando

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij} k_1) - M(1 - c_1) \leq c_m$$

se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

F.O.

c_m

d_3

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij} k_2) + f_2 - M(1 - c_2) \leq c_m$$

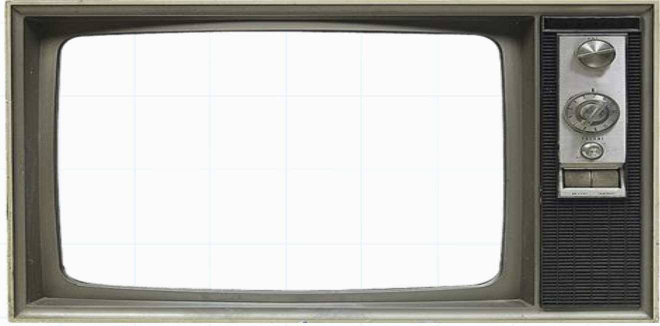
se contrato 2 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

contrato 3: determina valor da variável d3 primeiro (L.I.) (x e d3)

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

$d_3 \in \mathbb{R}^+$: distância percorrida no contrato 3, além de D_3 o que vai ser cobrado a mais

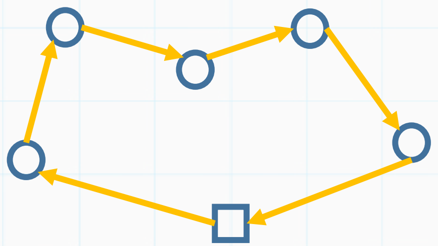
Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

Restrições:

Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.



$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)

Contrato 1: Existe um custo K_1 por km andando

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_1) - M(1 - c_1) \leq c_m$$

se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

F.O.

c_m

d_3

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_2) + f_2 - M(1 - c_2) \leq c_m$$

se contrato 2 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

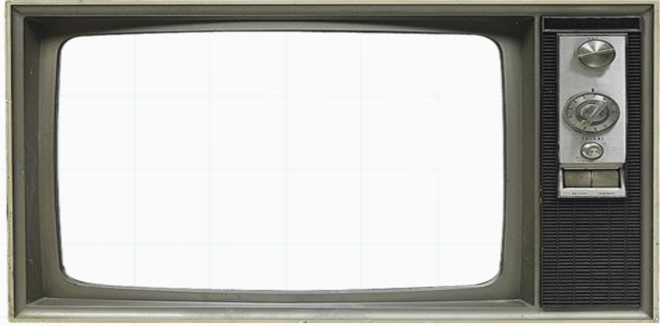
$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}) - D_3 \leq d_3$$

contrato 3: determina valor da variável d3 primeiro (L.I.) (x e d3)

$$x_{ij} = \begin{cases} 1, & \text{se o caixeiro faz o trajeto de } i \text{ para } j, \forall ij \in A \\ 0, & \text{caso contrário} \end{cases}$$

$d_3 \in \mathbb{R}^+$: distância percorrida no contrato 3, além de D_3 o que vai ser cobrado a mais

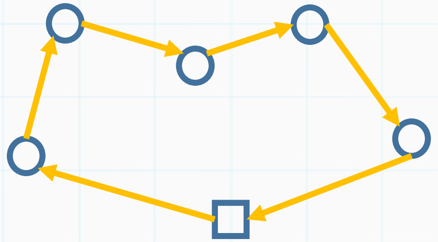
Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

Restrições:

Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.



$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)

Contrato 1: Existe um custo K_1 por km andando

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_1) - M(1 - c_1) \leq c_m$$

se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_2) + f_2 - M(1 - c_2) \leq c_m$$

se contrato 2 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

contrato 3: determina valor da variável d3 primeiro (L.I.) (x e d3)

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}) - D_3 \leq d_3$$

se contrato 3 ativo então seu custo é um L.I. para a F.O. (em função de d3) (d3, c, e c_m)

F.O.

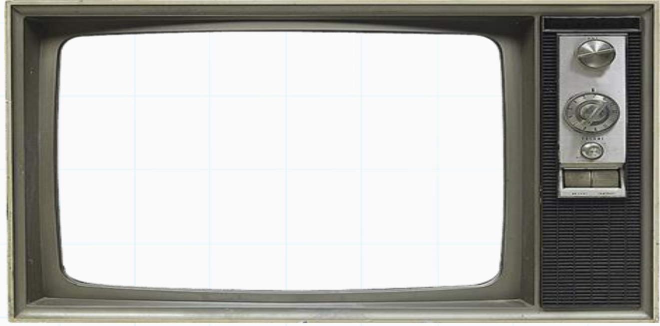
$\downarrow$

c_m

$\downarrow$

d_3

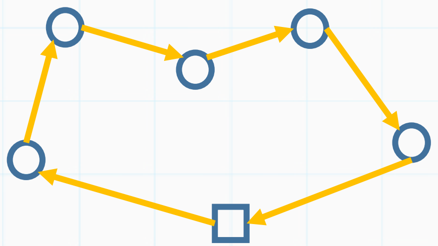
Formulações Clássicas



Problema do Caixeiro Viajante com contratos:

Restrições:

Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.



$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)

Contrato 1: Existe um custo K_1 por km andando

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_1) - M(1 - c_1) \leq c_m$$

se contrato 1 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_2) + f_2 - M(1 - c_2) \leq c_m$$

se contrato 2 ativo então seu custo é um L.I. para a F.O. (x, c e c_m)

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}) - D_3 \leq d_3$$

contrato 3: determina valor da variável d3 primeiro (L.I.) (x e d3)

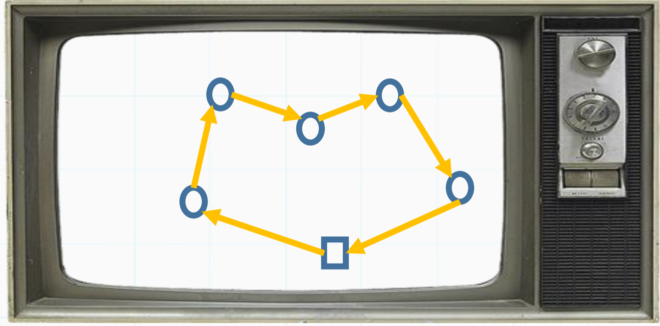
$$d_3k_3 + f_3 - M(1 - c_3) \leq c_m$$

se contrato 3 ativo então seu custo é um L.I. para a F.O. (em função de d3) (d3, c, e c_m)

$$\text{F.O.} \downarrow c_m \downarrow d_3$$

Problema do Caixeiro Viajante com contratos:

Formulações Clássicas



$$\min \quad c_m$$

$$\sum_{\substack{j \in V \\ i \neq j}} x_{ij} = 1, \quad \forall i \in V$$

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1, \quad \forall j \in V$$

$$\sum_{\substack{i,j \in S \\ i \neq j}} x_{ij} \leq |S| - 1, \quad \forall S \subset V, |S| \geq 2$$

Caixeiro clássico

$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo (c)

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij} k_1) - M(1 - c_1) \leq c_m$$

se contrato 1 ativo então seu custo é um L.I. para a F.O.

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij} k_2) + f_2 - M(1 - c_2) \leq c_m$$

se contrato 2 ativo então seu custo é um L.I. para a F.O

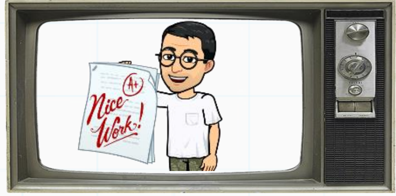
$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}) - D_3 \leq d_3$$

contrato 3: determina valor da variável d3 primeiro (L.I.)

$$d_3 k_3 + f_3 - M(1 - c_3) \leq c_m$$

se contrato 3 ativo então seu custo é um L.I. para a F.O. (em função de d3)

Exercício



Problema do Roteamento de Veículos Capacitado com contratos: Seja $G=(V,A)$ um grafo direcionado, $V=\{0,\dots,n\}$ (0 é depósito), onde cada vértice $i \in V \setminus \{0\}$ tem demanda q_i e cada arco (i,j) tem custo c_{ij}. A frota tem um conjunto K de motoristas que podem dirigir caminhões com capacidade Q.

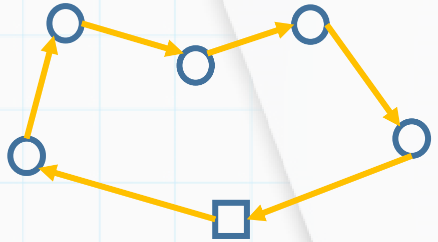
Roteamento clássico

Determinar $|K|$ rotas que atendam as demandas de todos os clientes onde o custo financeiro de contratação de caminhões é mínimo, dado que cada motorista pode usar um caminhão com uma das contratações listadas

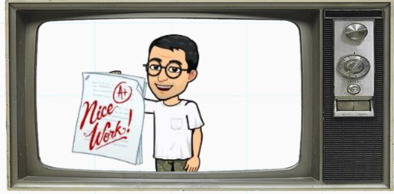
Contratos

Contrato 1: Existe um custo K_1 por km andando
Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado
Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.

Esse problema é a união do Problema de roteamento de veículos original com o Problema do Caixeiro Viajante com contratos, só que agora cada rota pode ter um contrato diferente.



Exercício



$$\min \sum_{k \in K} \sum_{ij \in A} c_{ij} x_{ijk}$$

$$\sum_{j \in N^-(i)} x_{ijk} - \sum_{j \in N^+(i)} x_{jik} = 0, \quad \forall k \in K, \forall i \in V$$

$$\sum_{\substack{ij \in A \\ i \neq 0}} q_i x_{ijk} \leq Q, \quad \forall k \in K$$

$$\sum_{ij \in A} c_{ij} x_{ijk} \leq L, \quad \forall k \in K$$

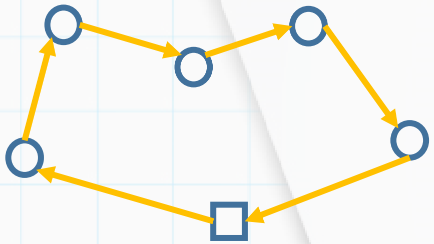
$$\sum_{k \in K} \sum_{j \in N^+(i)} x_{jik} = 1, \quad \forall i \in V \backslash \{0\}$$

$$\sum_{ij \in A[S]} x_{ijk} \leq |S| - 1, \quad \forall S \subseteq V \backslash \{0\}, 2 \leq |S|, k \in K$$

$$\sum_{j \in N^-(0)} x_{0jk} \leq 1, \quad \forall k \in K$$

<u>modelo clássico de roteamento</u>:

1) Muda alguma coisa nas restrições da definição das rotas ?



Variáveis de contrato:

$$c_i = \begin{cases} 1, & \text{se o contrato } i \in \{1,2,3\} \text{ está ativo} \\ 0, & \text{caso contrário} \end{cases}$$

$$c_m \in \mathbb{R}^+ : \text{ custo financeiro}$$

$$d_3 \in \mathbb{R}^+ : \text{ distância percorrida no contrato 3, além de } D_3 \text{ o que vai ser cobrado a mais}$$

Restrições de contrato:

$$\sum_{i=1}^{3} c_i = 1$$

só podemos ter um contrato ativo

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_1) - M(1 - c_1) \leq c_m$$

contrato 1

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}k_2) + f_2 - M(1 - c_2) \leq c_m$$

contrato 2

$$\sum_{\substack{i,j \in V \\ i \neq j}} x_{ij}(c_{ij}) - D_3 \leq d_3$$

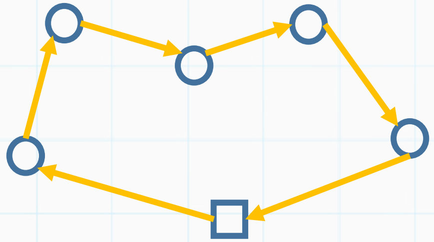
d3 do contrato 3

$$d_3k_3 + f_3 - M(1 - c_3) \leq c_m$$

Contrato 3

Contrato 1: Existe um custo K_1 por km andando
Contrato 2: Tem um custo fixo F_2 pelo contrato, e um custo K_2 por km andado
Contrato 3: Tem um custo fixo F_3 pelo contrato, e um custo K_3 por km andado a mais que a distância D_3.

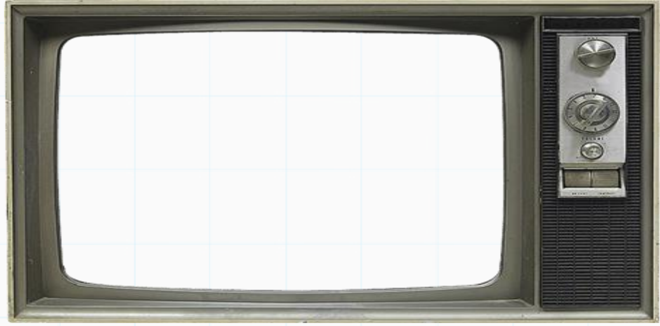
Exercício



1) Vars: as variáveis de contrato irão ganhar um índice a mais (para cada caminhão), formalize essa definição.
2) Rest: reescreva cada uma das 5 restrições de contrato, levando em consideração essa expansão das variáveis.

(SÓ PRECISA ESCREVER AS RESTRIÇÕES DE CONTRATO)

Formulações Clássicas

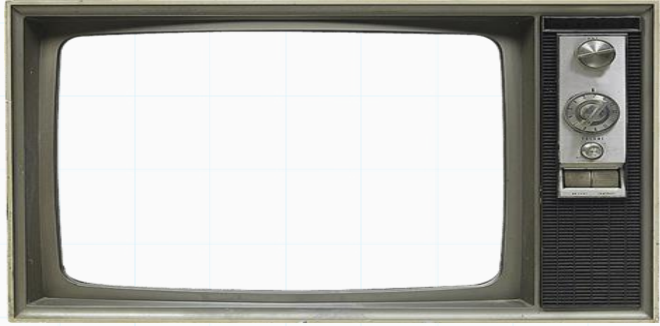


Problema do escalonamento de tarefas:

- Conjunto J de tarefas

J1
J2
J3

Formulações Clássicas



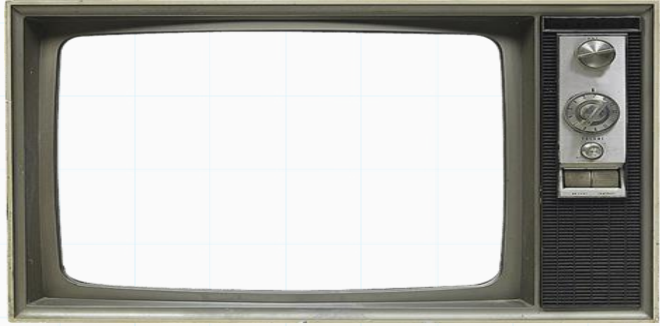
Problema do escalonamento de tarefas:

- Conjunto J de tarefas
- Conjunto M de máquinas

J1
J2
J3

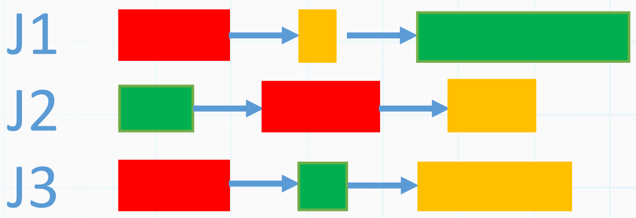
M1
M2
M3

Formulações Clássicas



<u>Problema do escalonamento de tarefas</u>:

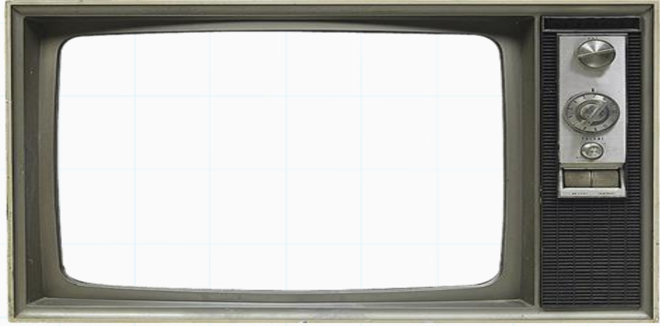
- Conjunto J de tarefas
- Conjunto M de máquinas
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \ldots, \phi_m^j)$ de processamento nas máquinas



Linha de produção -> produto para estar pronto tem que passar por várias máquinas (Ex: Carro na linha de montagem)

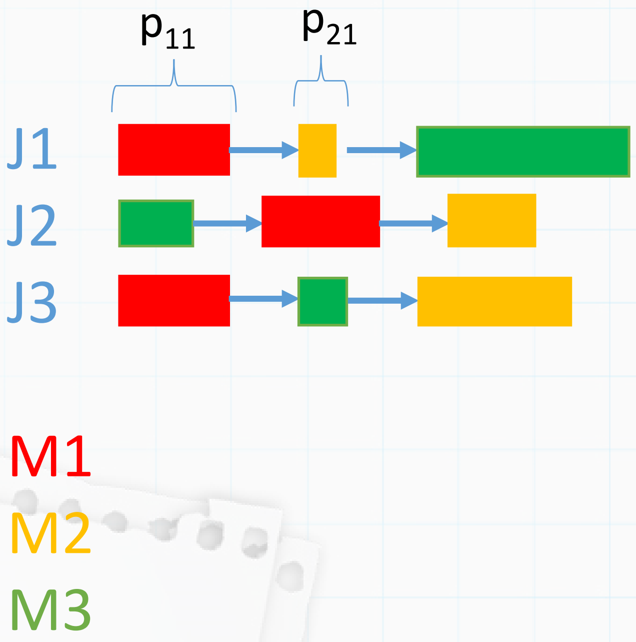
M1
M2
M3

Formulações Clássicas



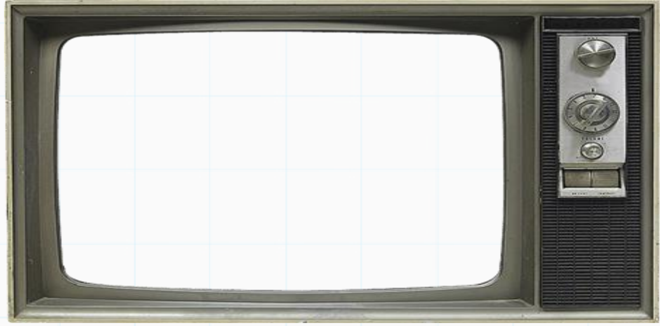
Problema do escalonamento de tarefas:

- Conjunto J de tarefas
- Conjunto M de máquinas
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \dots, \phi_m^j)$ de processamento nas máquinas
- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}



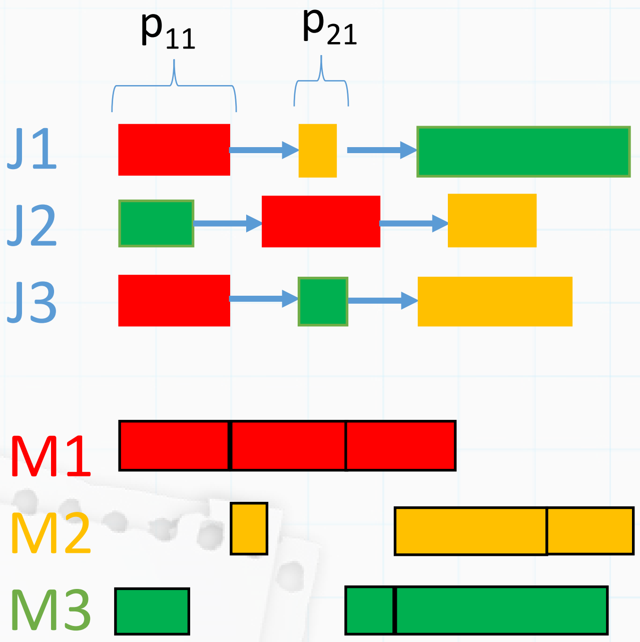
Linha de produção -> produto para estar pronto tem que passar por várias máquinas (Ex: Carro na linha de montagem)

Formulações Clássicas



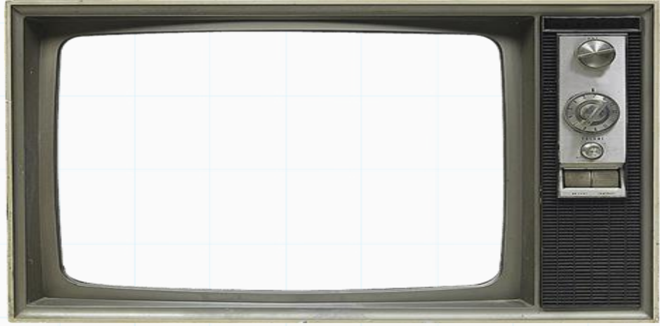
Problema do escalonamento de tarefas:

- Conjunto J de tarefas
- Conjunto M de máquinas
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \dots, \phi_m^j)$ de processamento nas máquinas
- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}
- Cada máquina só processa uma coisa por vez (sem paralelismo), e uma vez começado o processamento da tarefa na máquina, ele vai até o fim.



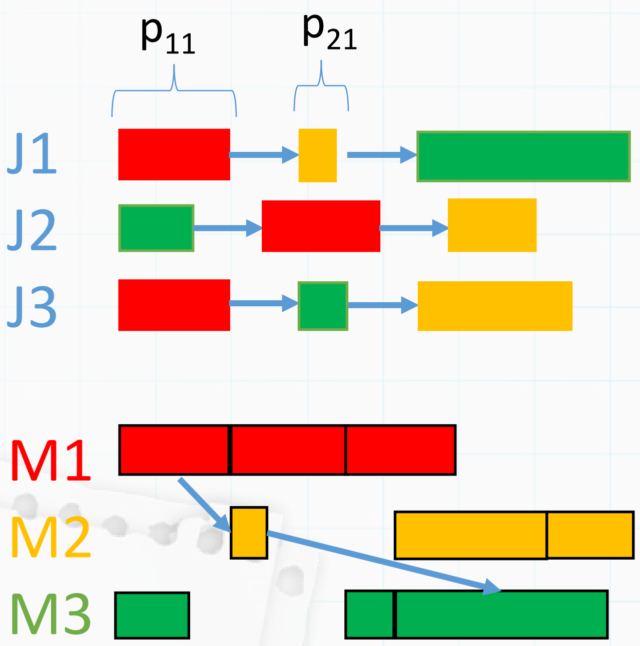
Linha de produção -> produto para estar pronto tem que passar por várias máquinas (Ex: Carro na linha de montagem)

Formulações Clássicas



<u>Problema do escalonamento de tarefas</u>:

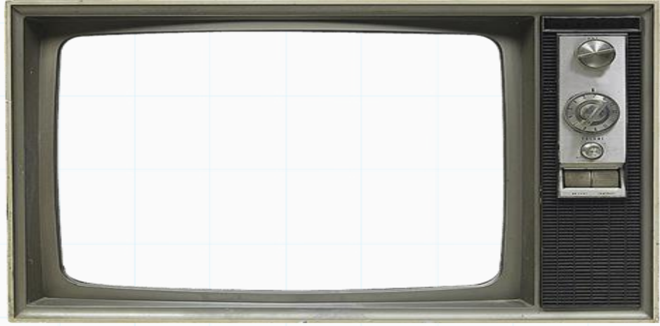
- Conjunto J de tarefas
- Conjunto M de máquinas
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \ldots, \phi_m^j)$ de processamento nas máquinas
- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}
- Cada máquina só processa uma coisa por vez (sem paralelismo), e uma vez começado o processamento da tarefa na máquina, ele vai até o fim.



Linha de produção -> produto para estar pronto tem que passar por várias máquinas (Ex: Carro na linha de montagem)

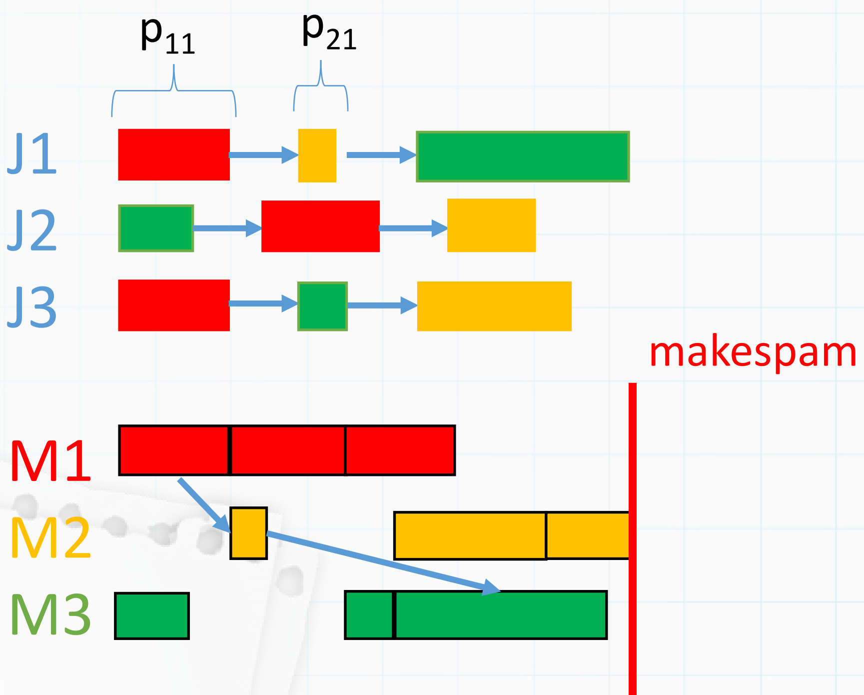
J1

Formulações Clássicas



Problema do escalonamento de tarefas:

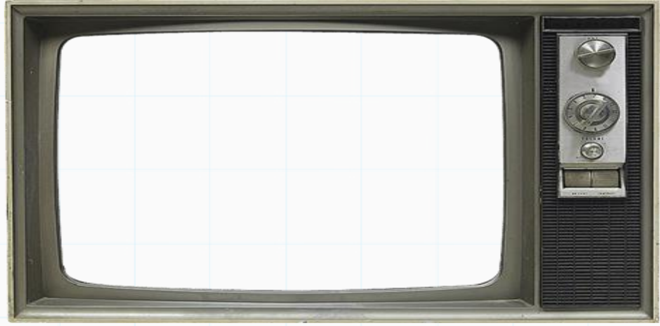
- Conjunto J de tarefas
- Conjunto M de máquinas
- Cada tarefa $j \in J$ tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \dots, \phi_m^j)$ de processamento nas máquinas
- Cada tarefa $j \in J$ e máquina $i \in M$ possui um tempo de processamento p_{ij}
- Cada máquina só processa uma coisa por vez (sem paralelismo), e uma vez começado o processamento da tarefa na máquina, ele vai até o fim.
- O objetivo é encontrar um escalonamento que minimize o makespam



Linha de produção -> produto para estar pronto tem que passar por várias máquinas (Ex: Carro na linha de montagem)

J1

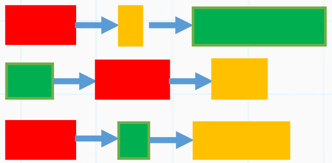
Formulações Clássicas



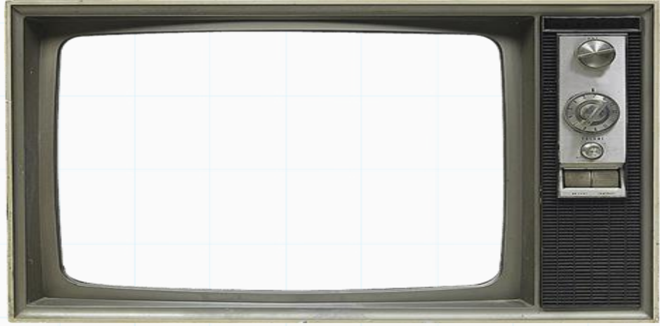
Problema do escalonamento de tarefas:

Variáveis:

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

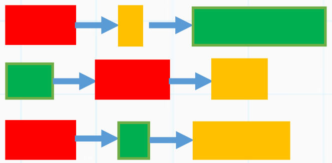


Formulações Clássicas



Problema do escalonamento de tarefas:

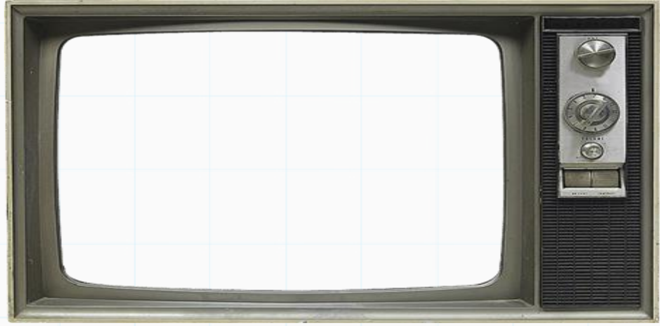
Variáveis:



$$x_{ij} \in \mathbb{R}^{+} = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

$$c_{max} \in \mathbb{R}^{+} = \{ \text{ makespam (término da última tarefa) } \}$$

Formulações Clássicas

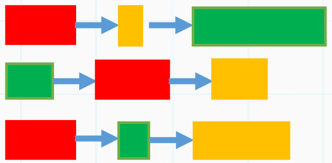


Problema do escalonamento de tarefas:

Variáveis:

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

$$c_{max} \in \mathbb{R}^+ = \{ \text{ makespam (término da última tarefa) } \}$$

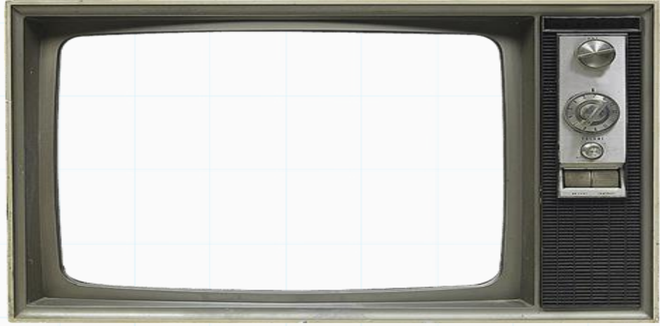


$$z_{ijk} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ precede a tarefa } k \in J \\ & \text{na máquina } i \in M \\ 0, & \text{caso contrário} \end{cases}$$

Precisamos para não deixar uma tarefa ser executada em cima da outra

Vamos definir as variáveis apenas para todo j < k, para evitar simetria

Formulações Clássicas

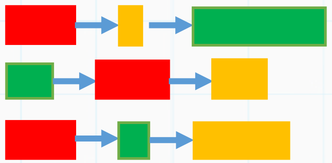


Problema do escalonamento de tarefas:

Variáveis:

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

$$c_{max} \in \mathbb{R}^+ = \{ \text{ makespam (término da última tarefa) } \}$$



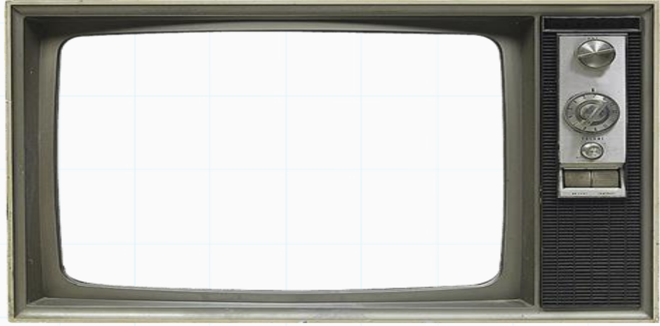
$$z_{ijk} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ precede a tarefa } k \in J \\ & \text{na máquina } i \in M \\ 0, & \text{caso contrário} \end{cases}$$

Precisamos para não deixar uma tarefa ser executada em cima da outra

Vamos definir as variáveis apenas para todo j < k, para evitar simetria

Função Objetivo:

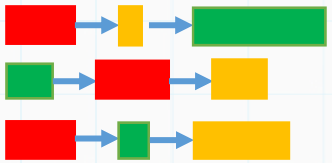
Formulações Clássicas



Problema do escalonamento de tarefas:

Variáveis:

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$



$$c_{max} \in \mathbb{R}^+ = \{\text{ makespam (término da última tarefa) }\}$$

$$z_{ijk} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ precede a tarefa } k \in J \\ & \text{na máquina } i \in M \\ 0, & \text{caso contrário} \end{cases}$$

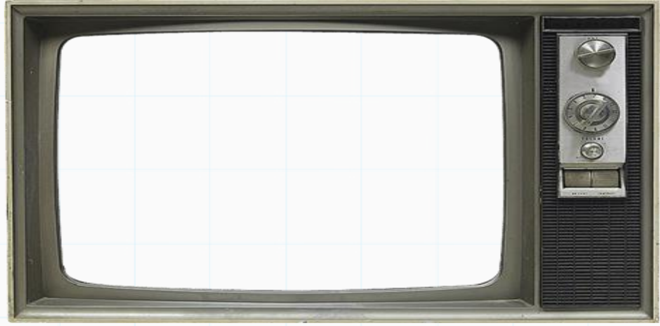
Precisamos para não deixar uma tarefa ser executada em cima da outra

Função Objetivo:

Vamos definir as variáveis apenas para todo j < k, para evitar simetria

$$\min c_{max}$$

Formulações Clássicas

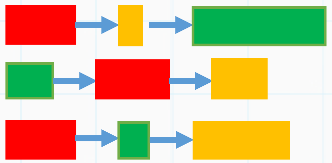


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi^j_i j} \geq \qquad \forall j \in J, \forall i = 2..., |M|$$

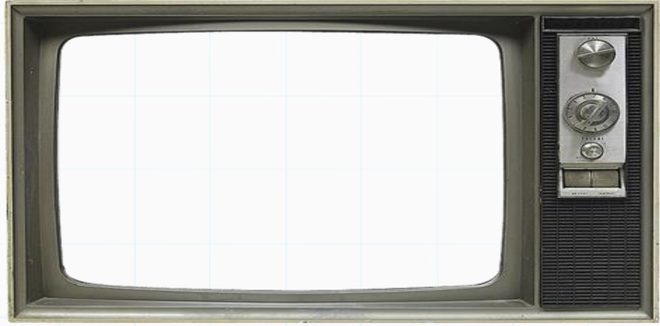
Precedência: se a tarefa foi executado num tempo x, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a x mais seu tempo de execução (x)



- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi^j_1, \phi^j_2, \ldots, \phi^j_m)$ de processamento nas máquinas

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

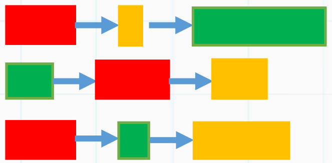


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2..., |M|$$

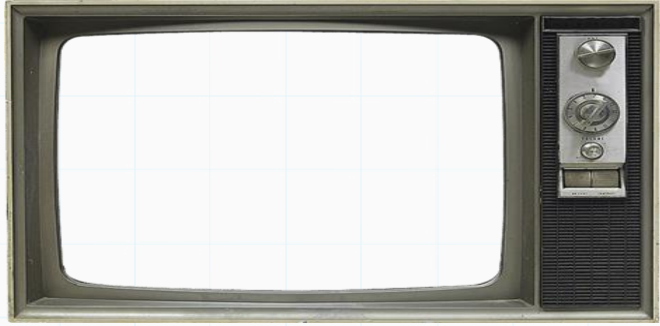
Precedência: se a tarefa foi executado num tempo x, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a x mais seu tempo de execução (x)



- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \dots, \phi_m^j)$ de processamento nas máquinas

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

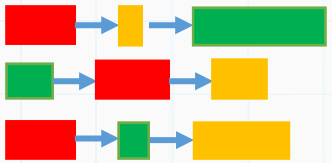


Problema do escalonamento de tarefas:

Restrições:

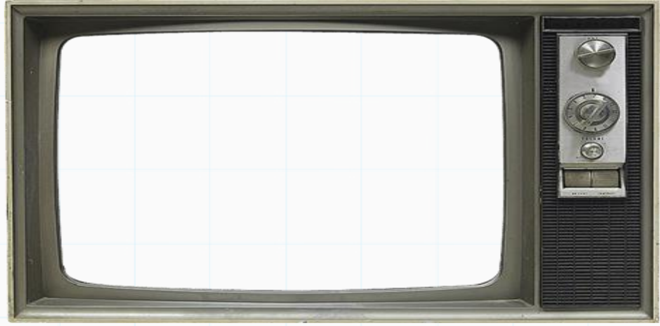
$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



em cada unidade de tempo na máquina, só pode estar executando uma tarefa

Formulações Clássicas

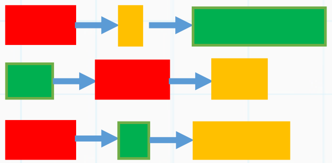


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

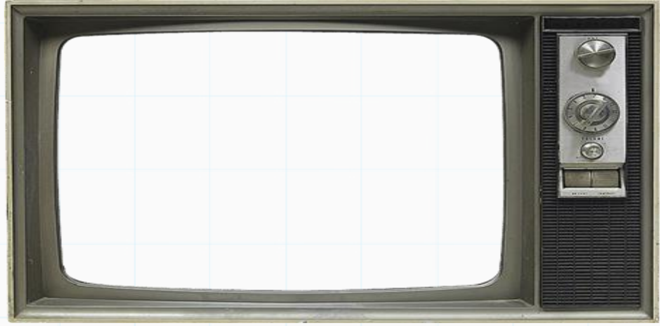
$$\forall j, k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

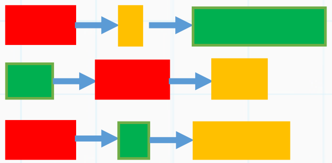


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi^j_i j} \geq x_{\phi^j_{i-1} j} + p_{\phi^j_{i-1} j}, \quad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

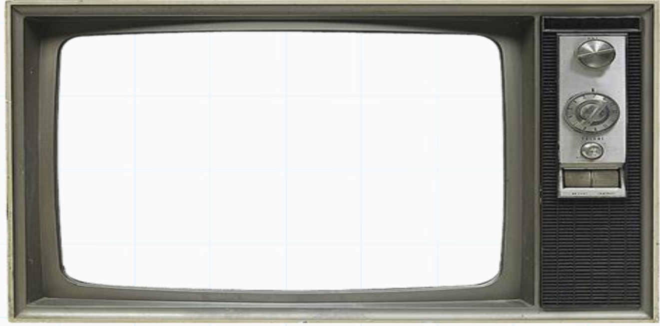
$$x_{ij} \geq x_{ik} + p_{ik} \qquad \forall j, k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

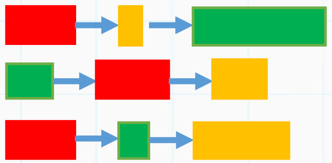


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi^j_i j} \geq x_{\phi^j_{i-1} j} + p_{\phi^j_{i-1} j}, \quad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \qquad \forall j, k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

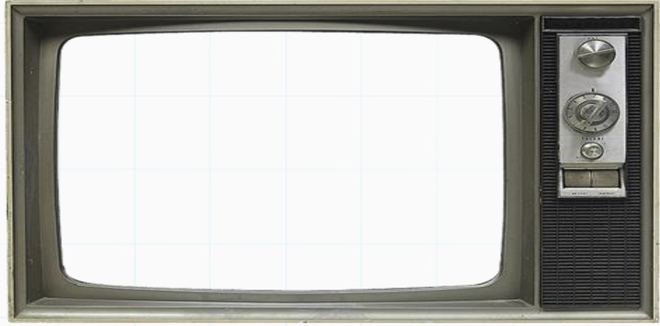
$$\forall j, k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

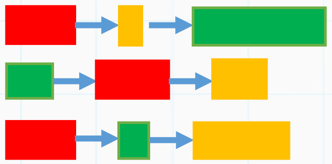


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi^j_i j} \geq x_{\phi^j_{i-1} j} + p_{\phi^j_{i-1} j}, \quad \forall j \in J, \forall i = 2\ldots, |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \qquad \forall j, k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

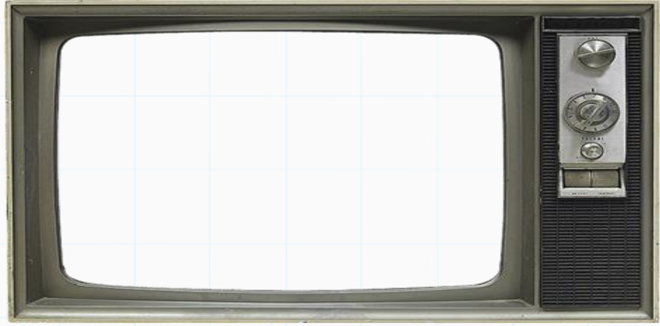
$$x_{ik} \geq x_{ij} + p_{ij} \qquad \forall j, k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

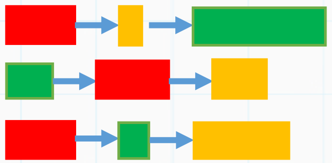


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi^j_i j} \geq x_{\phi^j_{i-1} j} + p_{\phi^j_{i-1} j}, \qquad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \qquad \forall j, k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

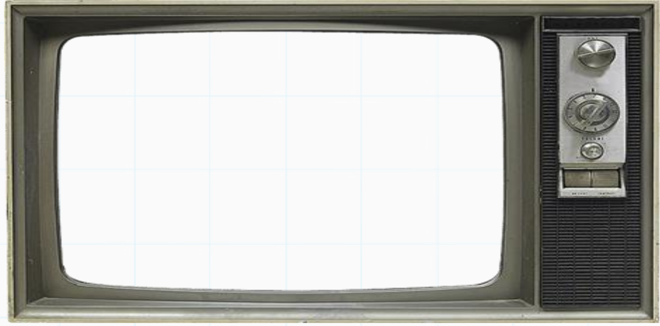
$$x_{ik} \geq x_{ij} + p_{ij} \qquad \forall j, k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$z_{ijk} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ precede a tarefa } k \in J \\ & \text{na máquina } i \in M \\ 0, & \text{caso contrário} \end{cases}$$

Usar variáveis z para deixar ativa apenas uma das duas restrições

Formulações Clássicas

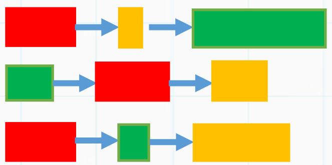


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi^j_i j} \geq x_{\phi^j_{i-1} j} + p_{\phi^j_{i-1} j}, \quad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \quad - BIGM(z_{ijk}), \quad \forall j, k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

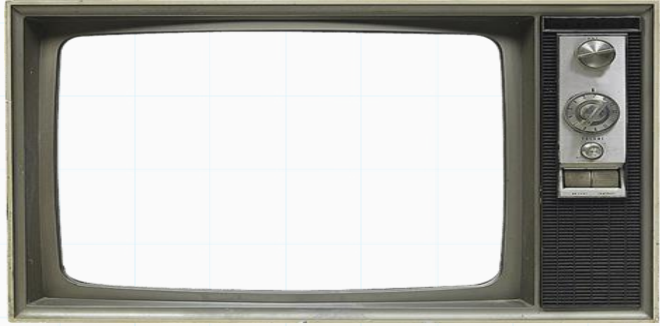
$$x_{ik} \geq x_{ij} + p_{ij} \qquad \forall j, k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$z_{ijk} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ precede a tarefa } k \in J \\ & \text{na máquina } i \in M \\ 0, & \text{caso contrário} \end{cases}$$

Usar variáveis z para deixar ativa apenas uma das duas restrições

Formulações Clássicas

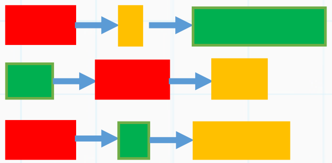


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \;\; - BIGM(z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

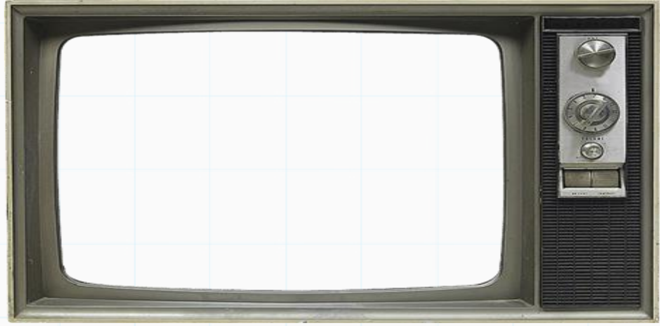
$$x_{ik} \geq x_{ij} + p_{ij} - BIGM(1 - z_{ijk}), \; \forall j,k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$z_{ijk} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ precede a tarefa } k \in J \\ & \text{na máquina } i \in M \\ 0, & \text{caso contrário} \end{cases}$$

Usar variáveis z para deixar ativa apenas uma das duas restrições

Formulações Clássicas

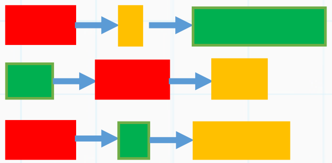


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2\ldots,|M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} - BIGM(z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

$$x_{ik} \geq x_{ij} + p_{ij} - BIGM(1 - z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

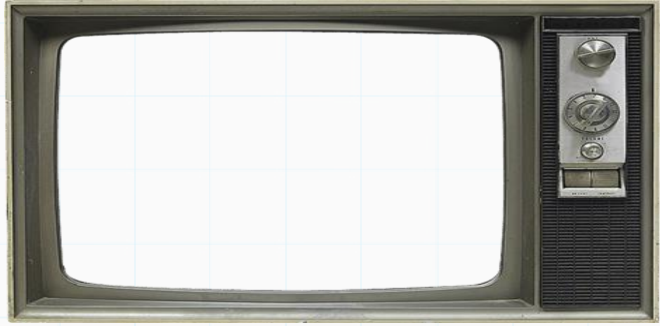
$$c_{max} \geq \qquad\qquad \forall j \in J$$

Makespam: L.I. dado pelo tempo de execução da tarefa na sua última máquina (x, c)

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}
- Cada tarefa j ∈ J tem uma ordem $\phi = (\phi_1^j, \phi_2^j, \ldots, \phi_m^j)$ de processamento nas máquinas

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

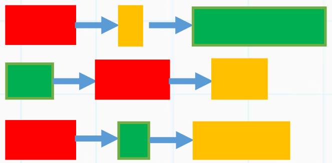


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2\ldots,|M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \; - BIGM(z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$x_{ik} \geq x_{ij} + p_{ij} - BIGM(1 - z_{ijk}), \; \forall j,k \in J, j < k, \forall i \in M$$

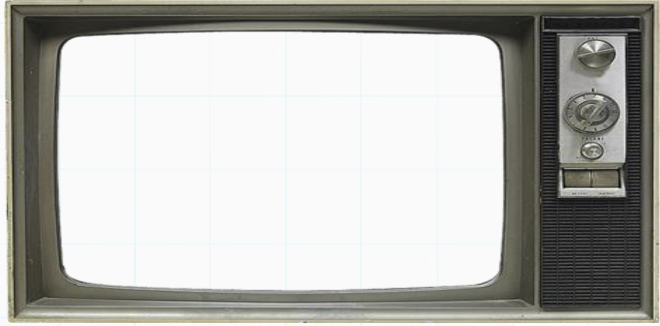
$$c_{max} \geq x_{\phi_m^j j} + p_{\phi_m^j j}, \quad \forall j \in J$$

Makespam: L.I. dado pelo tempo de execução da tarefa na sua última máquina (x, c)

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \ldots, \phi_m^j)$ de processamento nas máquinas

$$x_{ij} \in \mathbb{R}^+ = \{\text{início da execução da tarefa } j \in J \text{ na máquina } i \in M\}$$

Formulações Clássicas

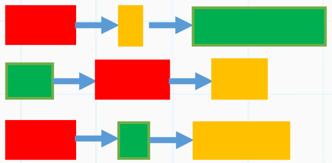


Problema do escalonamento de tarefas:

Restrições:

$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem (ϕ), ela tem que ser executado num tempo superior a t mais seu tempo de execução



se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \; - BIGM(z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

$$x_{ik} \geq x_{ij} + p_{ij} - BIGM(1 - z_{ijk}), \; \forall j,k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$c_{max} \geq x_{\phi_m^j j} + p_{\phi_m^j j}, \quad \forall j \in J$$

Makespam: L.I. dado pelo tempo de execução da tarefa na sua última máquina (x, c)

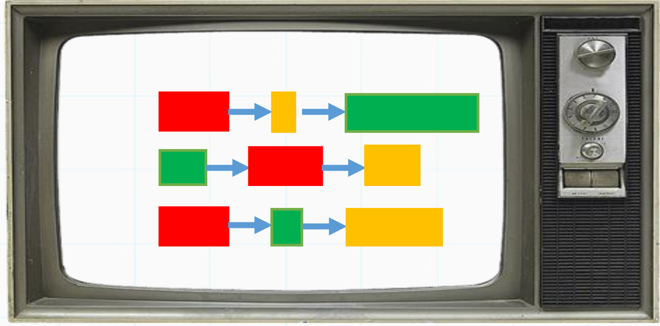
$$x_{ij} \geq 0, \quad \forall i \in M, \forall j \in J$$

$$z_{ijk} \in \{0,1\}, \quad \forall i \in M, \forall j,k \in J$$

integralidade e não negatividade

Problema do escalonamento de tarefas:

Formulações Clássicas



$$\min c_{max}$$

$$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2\ldots, |M|$$

Precedência

se tarefa k executa antes da tarefa j na máquina i (x)

$$x_{ij} \geq x_{ik} + p_{ik} \; - BIGM(z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

se tarefa j executa antes da tarefa k na máquina i (x)

$$x_{ik} \geq x_{ij} + p_{ij} - BIGM(1 - z_{ijk}), \; \forall j,k \in J, j < k, \forall i \in M$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$c_{max} \geq x_{\phi_m^j j} + p_{\phi_m^j j}, \quad \forall j \in J$$

Makespam

$$x_{ij} \geq 0, \quad \forall i \in M, \forall j \in J$$

$$z_{ijk} \in \{0,1\}, \quad \forall i \in M, \forall j,k \in J$$

integralidade e não negatividade

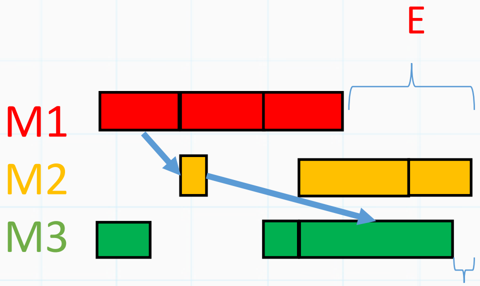
Exercício

Problema do escalonamento de tarefas com custo financeiro:

Considere o problema de escalonamento de tarefas apresentado nesta aula, mas agora cada máquina $i \in M$ possui um custo financeiro $cost_i$ por período de tempo usada (makespam), e queremos encontrar o escalonamento que minimize o custo pago pelo uso das maquinas (soma dos custos pagos).

e equilibrado:

Considere o problema anterior, só que agora, por uma questão operacional , os tempos de desligamento das máquinas (makespam) não podem ser muito distantes. Logo, a diferença de tempo do makespam entre quaisquer duas máquinas tem que ser no máximo E.



Lembrando:

- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \dots, \phi_m^j)$ de processamento nas máquinas
- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}

Modelo original

$$\min c_{max}$$

(1) $$x_{\phi_i^j j} \geq x_{\phi_{i-1}^j j} + p_{\phi_{i-1}^j j}, \quad \forall j \in J, \forall i = 2 \dots, |M|$$

(2) $$x_{ij} \geq x_{ik} + p_{ik} - BIGM(z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

(3) $$x_{ik} \geq x_{ij} + p_{ij} - BIGM(1 - z_{ijk}), \quad \forall j,k \in J, j < k, \forall i \in M$$

(4) $$c_{max} \geq x_{\phi_m^j j} + p_{\phi_m^j j}, \quad \forall j \in J$$

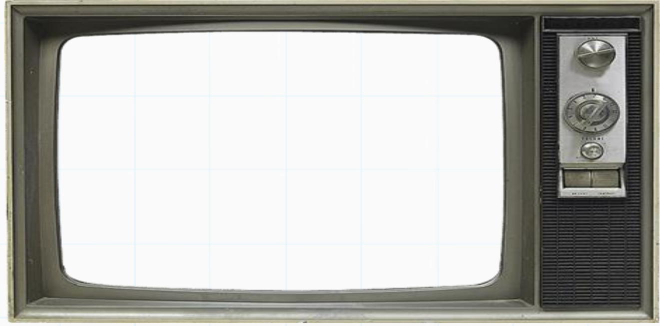
$$x_{ij} \geq 0, \quad \forall i \in M, \forall j \in J$$

$$z_{ijk} \in \{0,1\}, \quad \forall i \in M, \forall j,k \in J$$

1) Vars: substituir c_{max} por c_{max}^i (agora cada maquina tem um makespam)
2) F.O.: minimizar a soma dos custos, isto é, makespam de cada máquina x custo máquina.
3) Rest:
   - Adaptar restrição (4) para as novas variáveis c_{max}^i (pois precisamos de L.I. (limites inferiores)para essas variáveis.
   - Restrições de Equilíbrio : para cada par de máquinas, a diferença de tempo final de processamento (c_{max}^i) não pode ultrapassar E

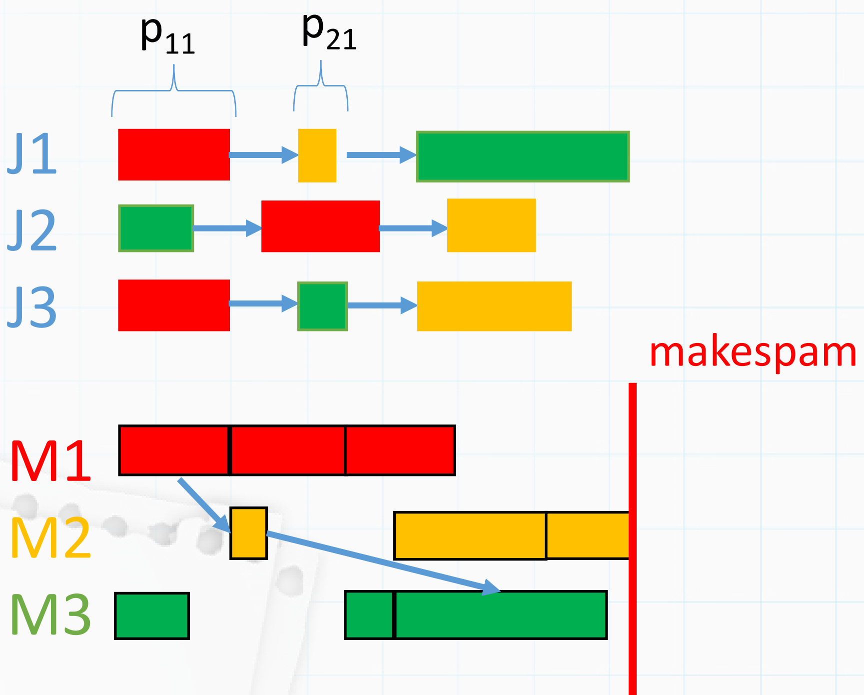
(ESCREVA APENAS A NOVA F.O. E AS NOVAS RESTRIÇÕES)

Formulações Clássicas

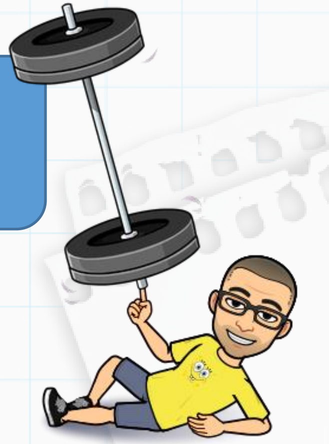


Problema do escalonamento de tarefas:

- Conjunto J de tarefas
- Conjunto M de máquinas
- Cada tarefa j ∈ J tem uma ordem $\phi=(\phi_1^j, \phi_2^j, \ldots, \phi_m^j)$ de processamento nas máquinas
- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}
- Cada máquina só processa uma coisa por vez (sem paralelismo), e uma vez começado o processamento da tarefa na máquina, ele vai até o fim.
- O objetivo é encontrar um escalonamento que minimize o makespam

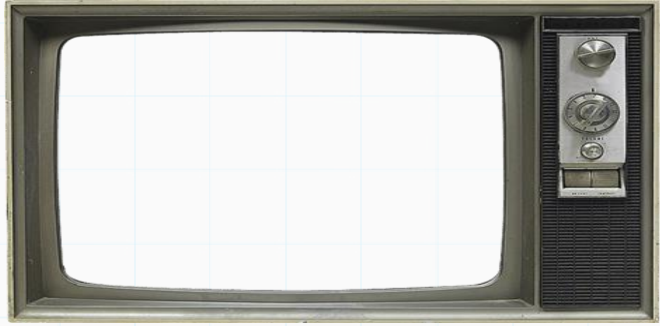


Formulação com variáveis binárias, mais complexo e mais forte

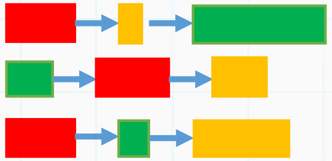


J1

Formulações Clássicas



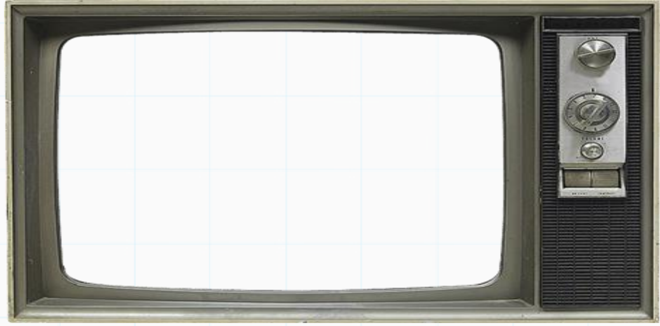
Problema do escalonamento de tarefas:



Variáveis:

$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in H \\ 0, & \text{caso contrário} \end{cases}$$

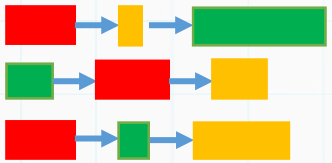
Formulações Clássicas



Problema do escalonamento de tarefas:

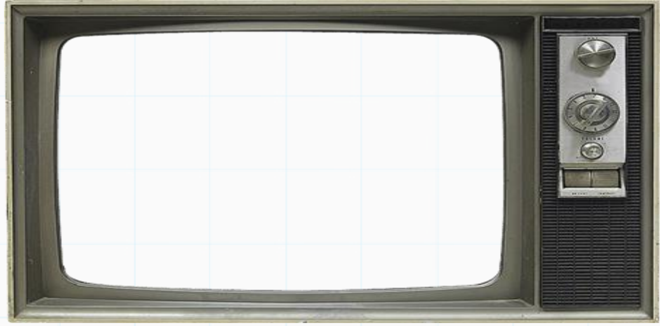
Variáveis:

$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in H \\ 0, & \text{caso contrário} \end{cases}$$

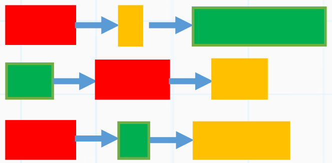


$$c_{max} \in \mathbb{R}^{+} = \{ \text{ makespam (término da última tarefa) } \}$$

Formulações Clássicas



Problema do escalonamento de tarefas:



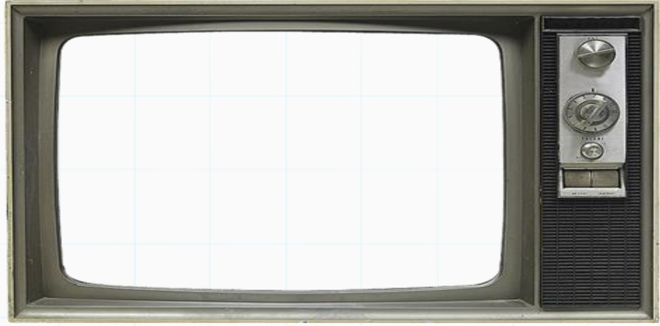
Variáveis:

$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in H \\ 0, & \text{caso contrário} \end{cases}$$

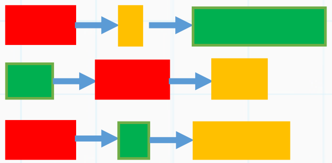
$$c_{max} \in \mathbb{R}^{+} = \{ \text{ makespam (término da última tarefa) } \}$$

Função Objetivo:

Formulações Clássicas



<u>Problema do escalonamento de tarefas</u>:



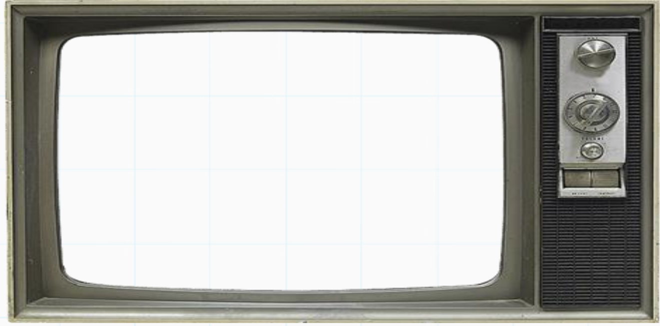
<u>Variáveis:</u>

$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in H \\ 0, & \text{caso contrário} \end{cases}$$

$$c_{max} \in \mathbb{R}^{+} = \{ \text{ makespam (término da última tarefa) } \}$$

<u>Função Objetivo:</u>

$$\min \; c_{max}$$



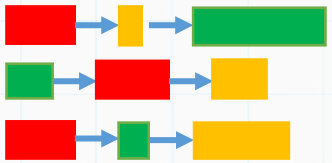
Formulações Clássicas

Problema do escalonamento de tarefas:

Restrições:

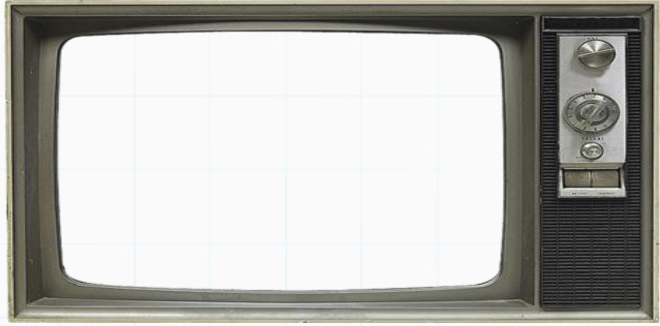
$$= 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in T \\ 0, & \text{caso contrário} \end{cases}$$

Formulações Clássicas

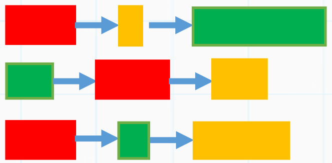


Problema do escalonamento de tarefas:

Restrições:

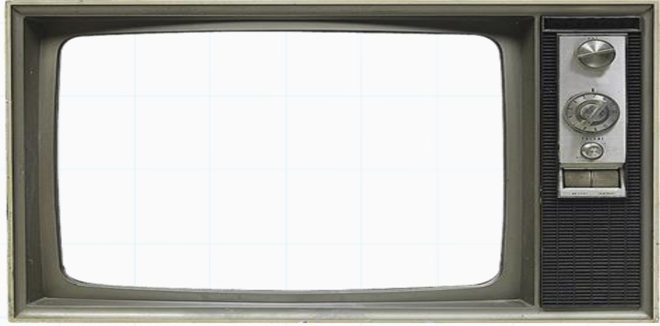
$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in T \\ 0, & \text{caso contrário} \end{cases}$$

Formulações Clássicas

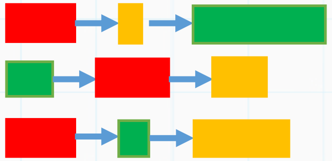


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\leq c_{max}, \quad \forall j \in J, \forall i \in M$$

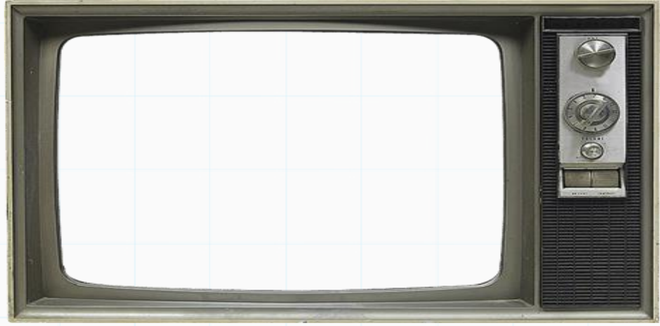
Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}

$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in T \\ 0, & \text{caso contrário} \end{cases}$$

$$c_{max} \in \mathbb{R}^+ = \{ \text{ makespam (término da última tarefa) } \}$$

Formulações Clássicas

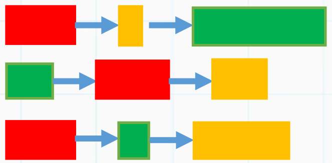


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

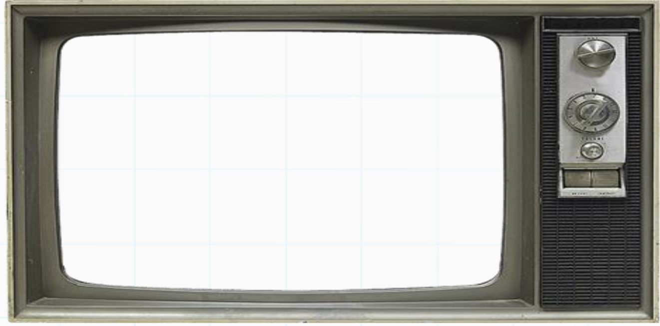
Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

- Cada tarefa j ∈ J e máquina i ∈ M possui um tempo de processamento p_{ij}

$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in T \\ 0, & \text{caso contrário} \end{cases}$$

$$c_{max} \in \mathbb{R}^{+} = \{ \text{ makespam (término da última tarefa) } \}$$

Formulações Clássicas

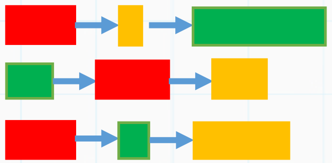


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)

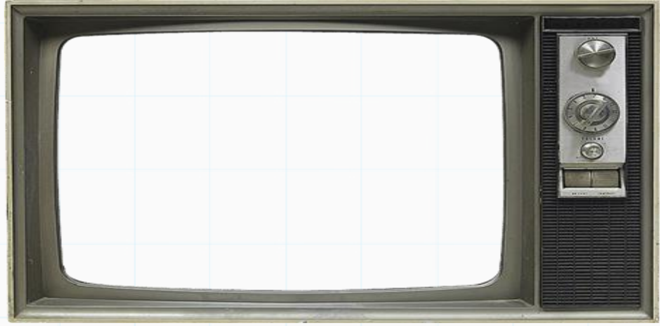


$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

Formulações Clássicas

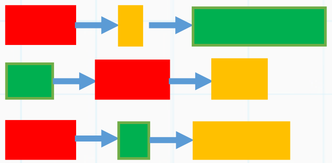


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

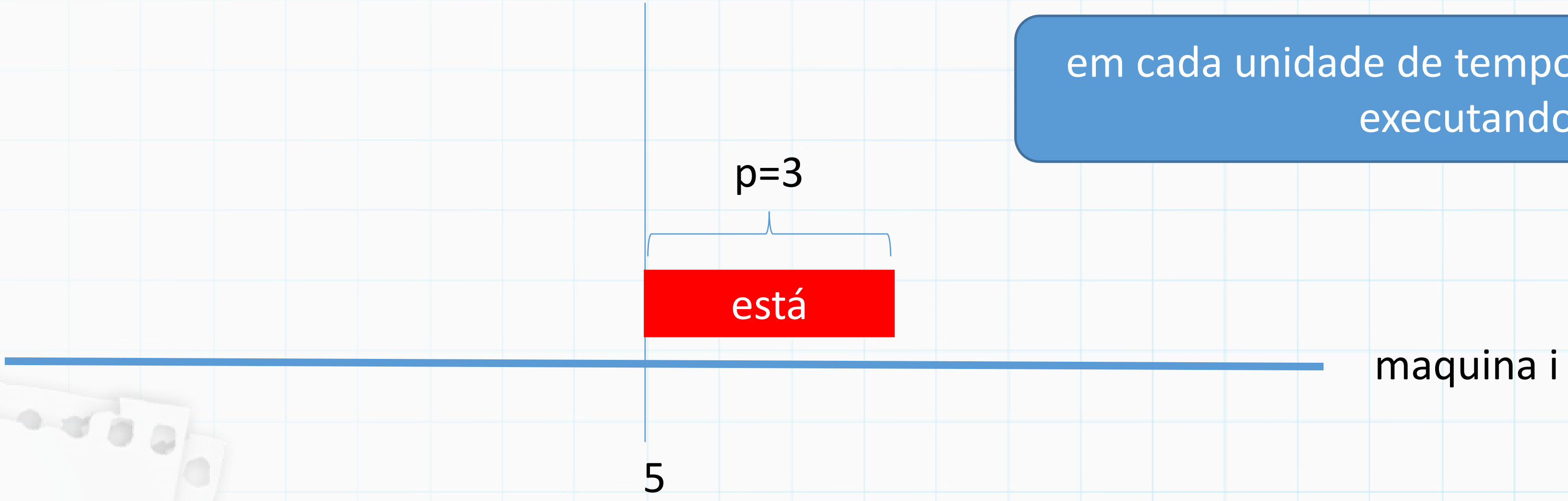
toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

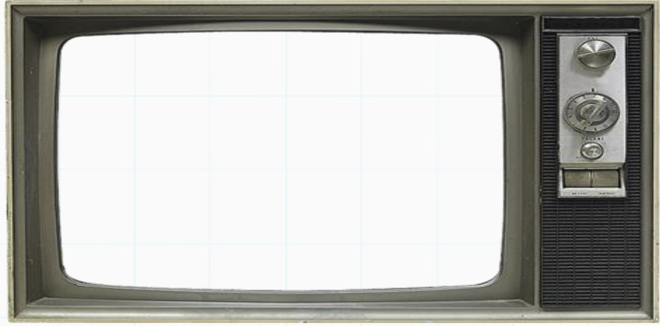
Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

em cada unidade de tempo na máquina, só pode estar executando uma tarefa



quem pode estar executando no tempo t ?

Formulações Clássicas

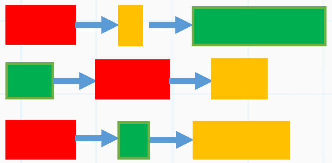


Problema do escalonamento de tarefas:

Restrições:

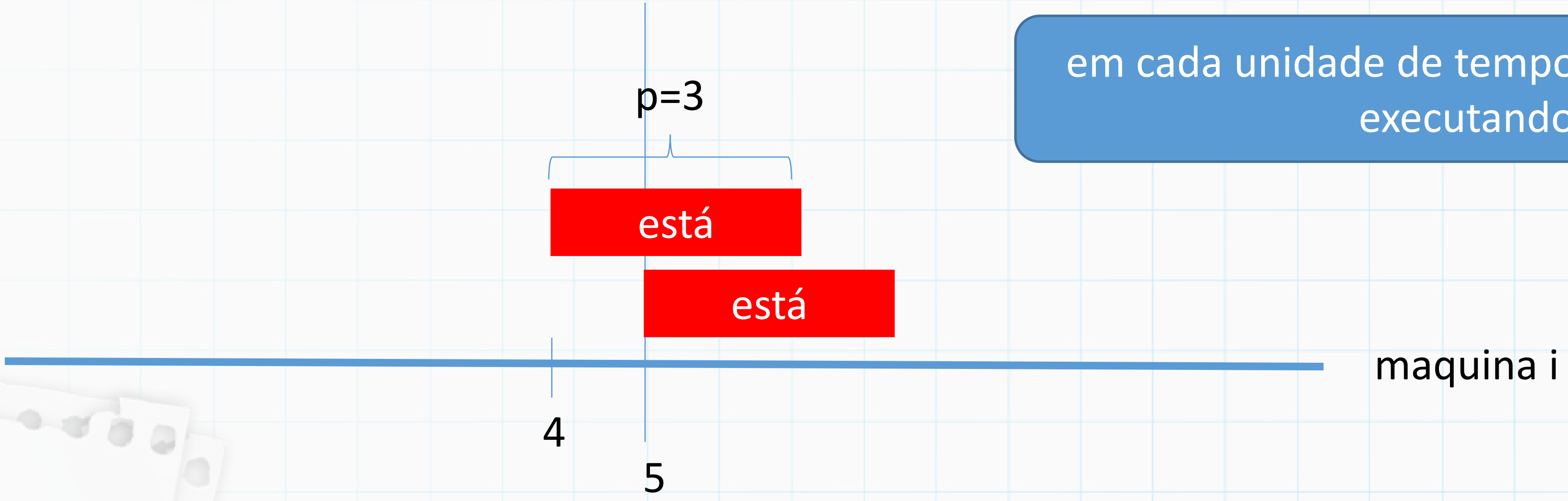
$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

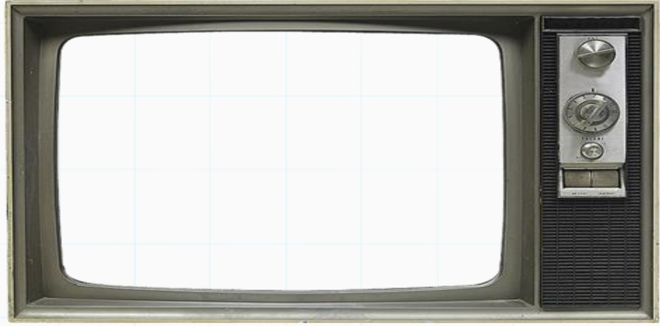
Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)



em cada unidade de tempo na máquina, só pode estar executando uma tarefa

quem pode estar executando no tempo t ?

Formulações Clássicas

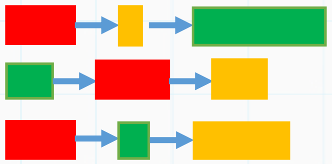


Problema do escalonamento de tarefas:

Restrições:

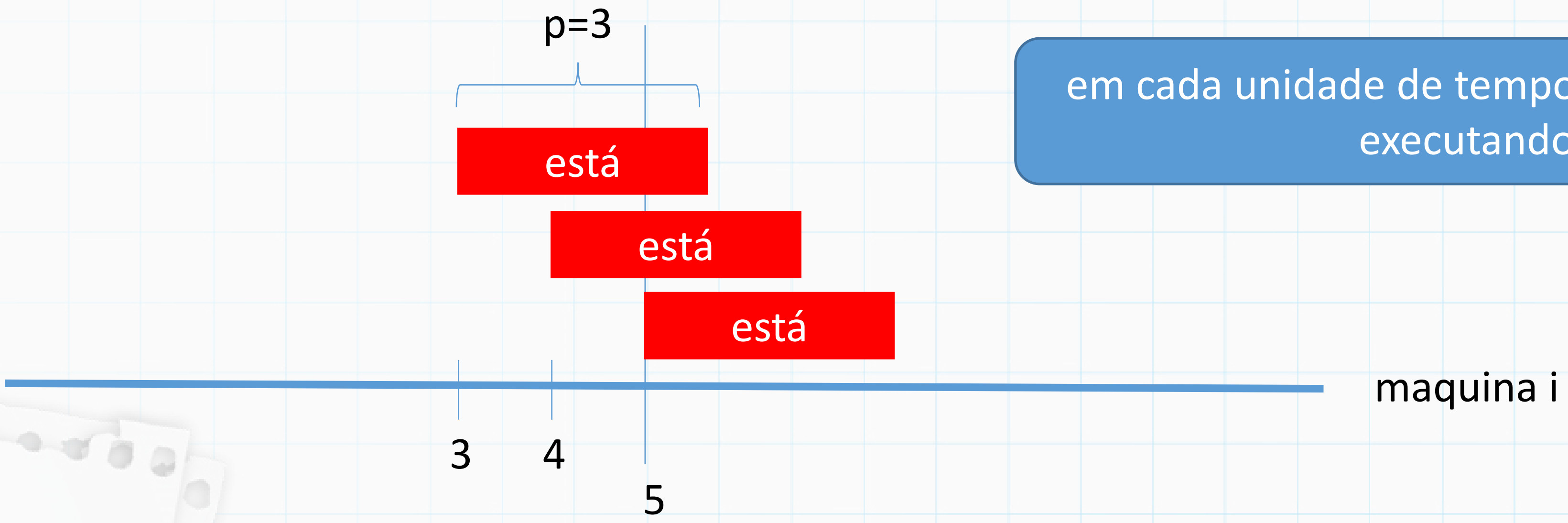
$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

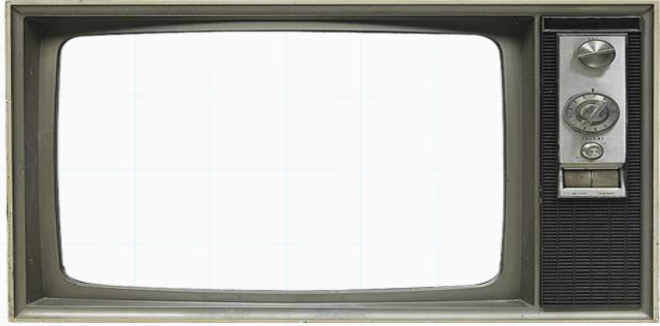
Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)



em cada unidade de tempo na máquina, só pode estar executando uma tarefa

quem pode estar executando no tempo t ?

Formulações Clássicas

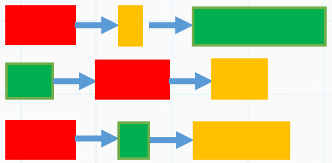


Problema do escalonamento de tarefas:

Restrições:

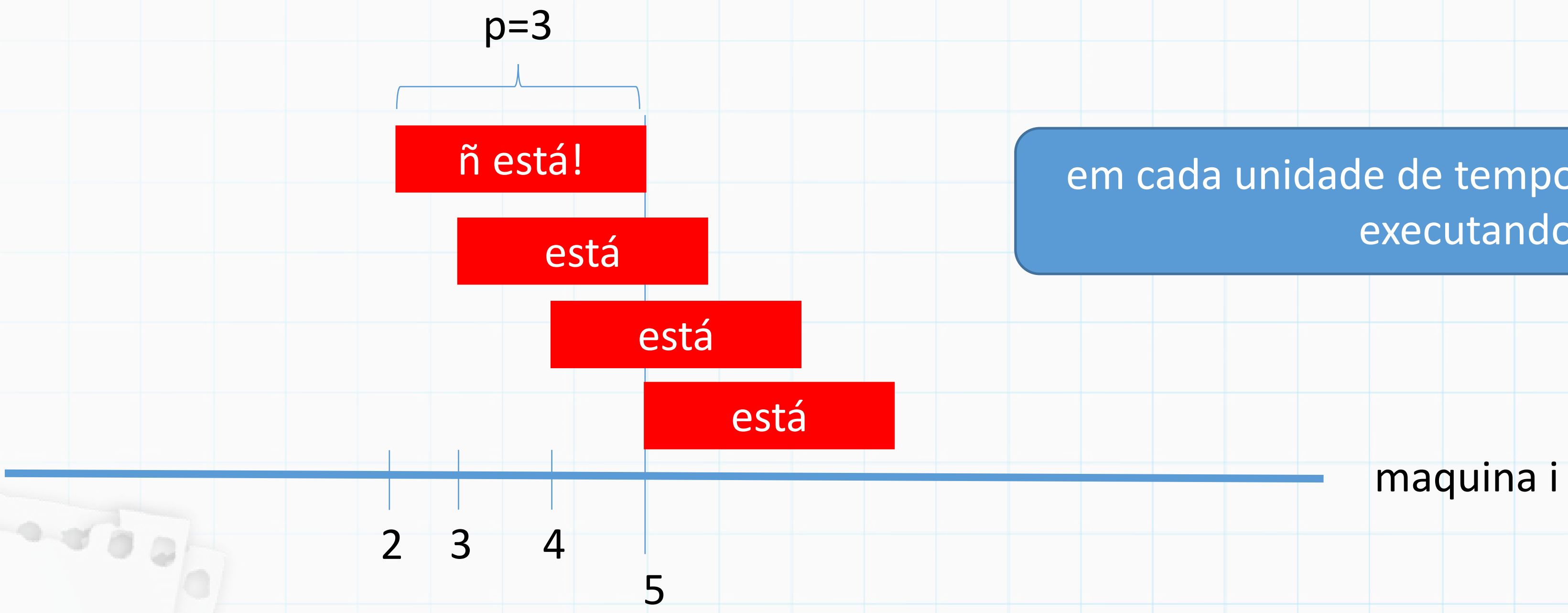
$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

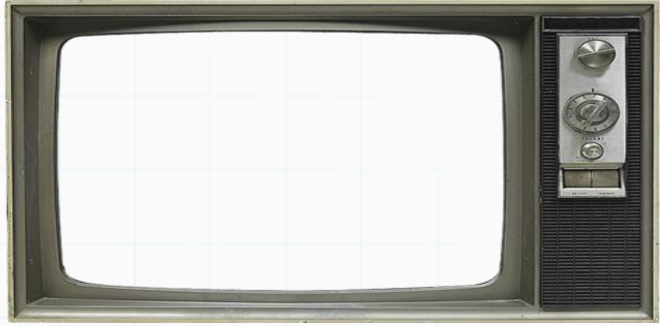
Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)



em cada unidade de tempo na máquina, só pode estar executando uma tarefa

quem pode estar executando no tempo t ?

Formulações Clássicas

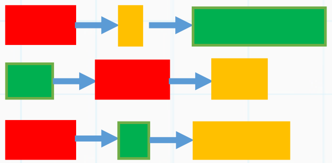


Problema do escalonamento de tarefas:

Restrições:

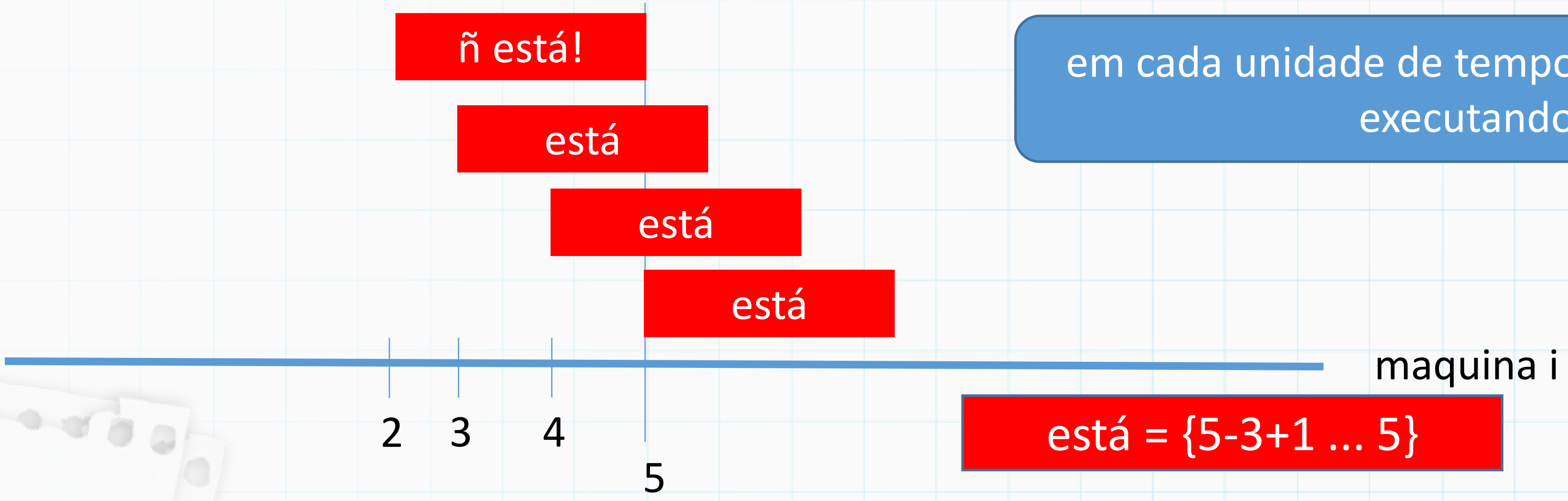
$$\sum_{t\in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t\in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

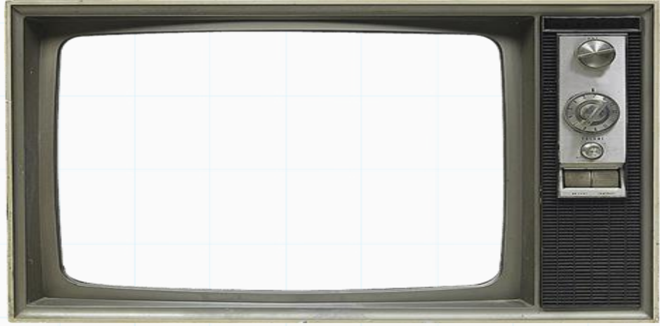
Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)



em cada unidade de tempo na máquina, só pode estar executando uma tarefa

quem pode estar executando no tempo t ?

Formulações Clássicas

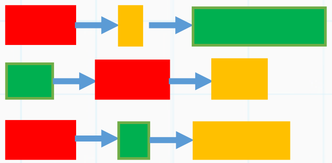


Problema do escalonamento de tarefas:

Restrições:

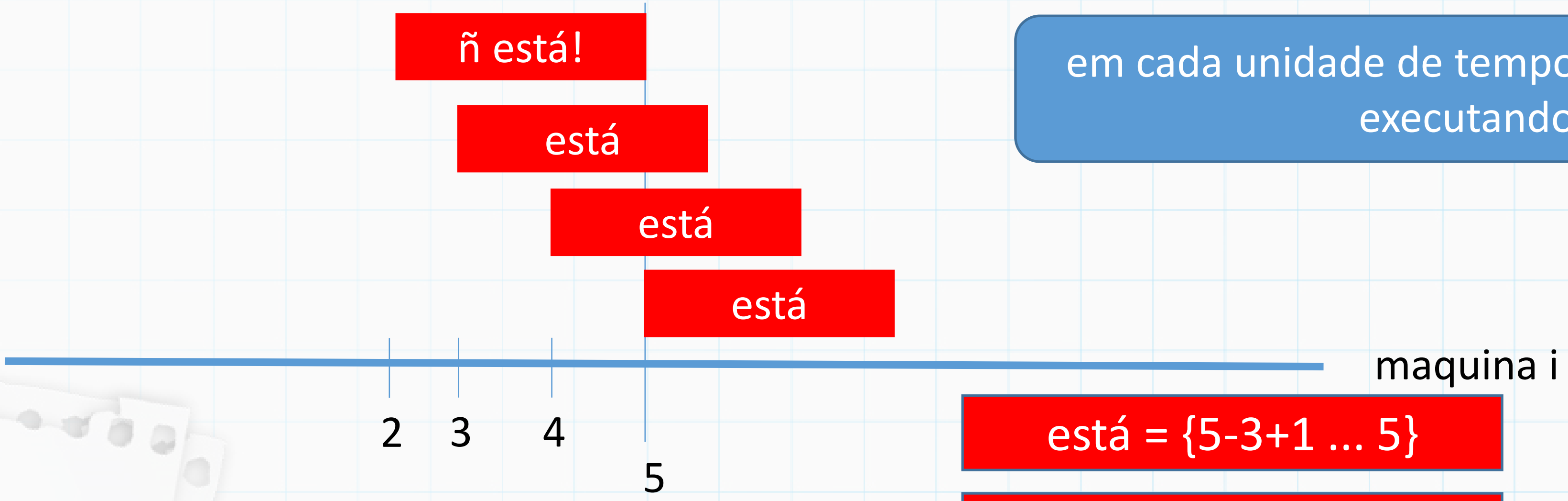
$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)



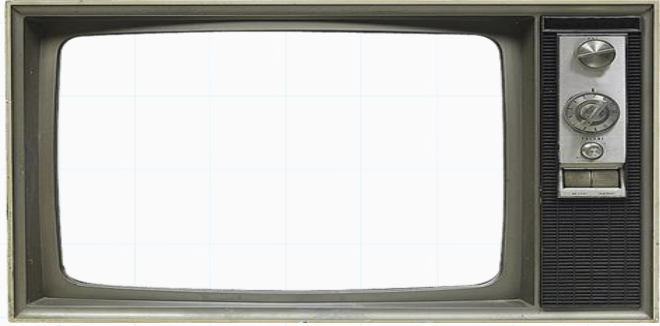
em cada unidade de tempo na máquina, só pode estar executando uma tarefa

quem pode estar executando no tempo t ?

está = {5-3+1 ... 5}

está = {t-p_{ij}+1 ... t}

Formulações Clássicas

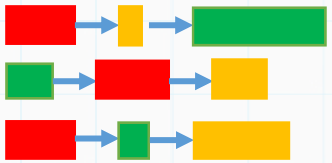


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)

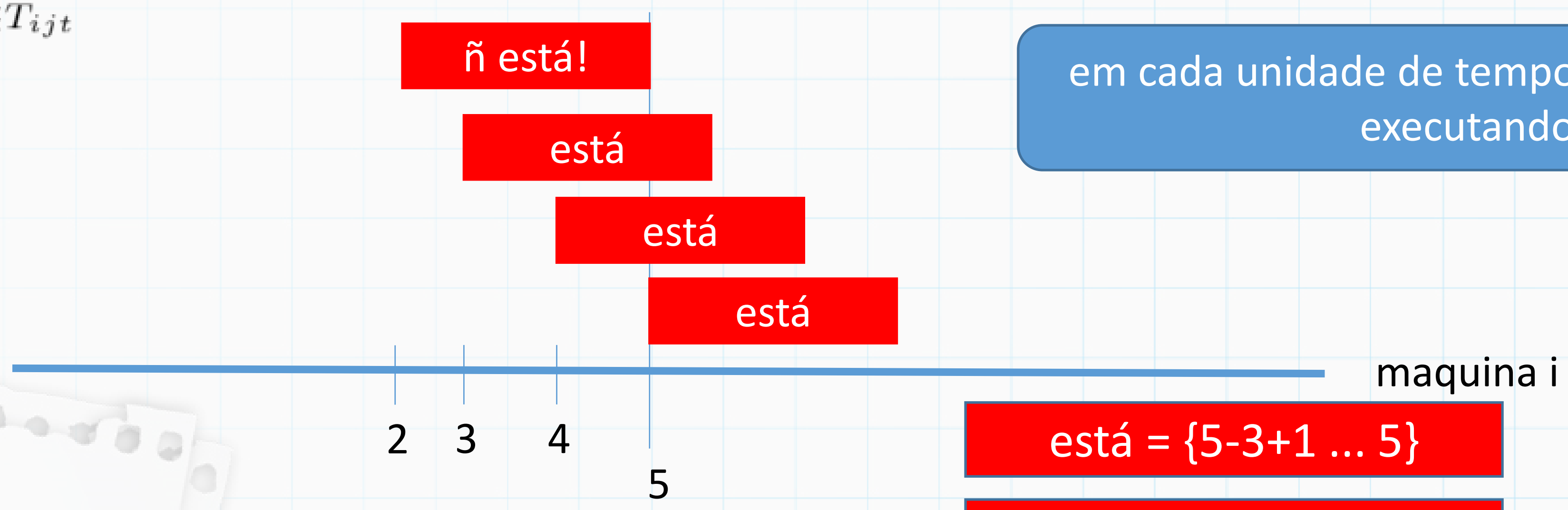


$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

$$\sum_{j \in J} \sum_{t' \in T_{ijt}} x_{ijt'} \leq 1, \quad \forall i \in M, \forall t \in H, \quad \text{onde } T_{ijt} = \{t - p_{ij} + 1, ..., t\}$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

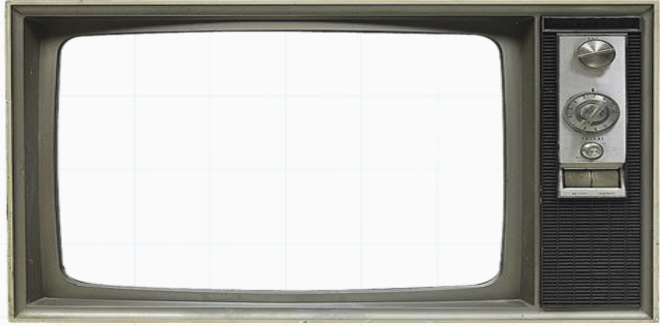


quem pode estar executando no tempo t ?

está = {5-3+1 ... 5}

está = {t-p_{ij}+1 ... t}

Formulações Clássicas

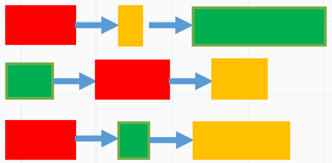


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t\in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)

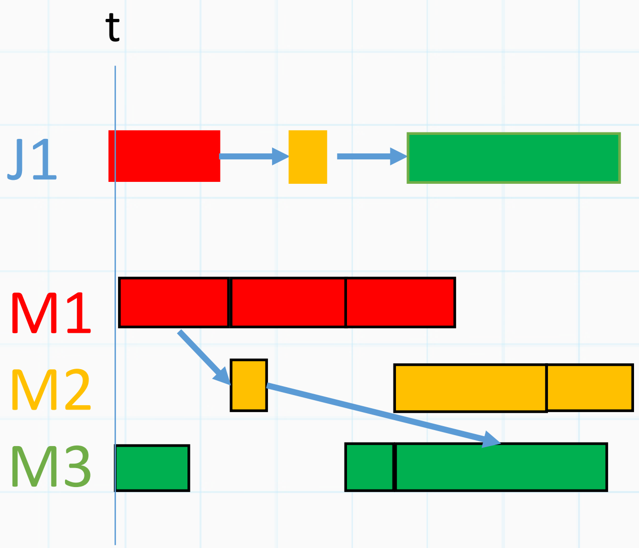


$$\sum_{t\in H} (t+p_{ij})x_{ijt} \le c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

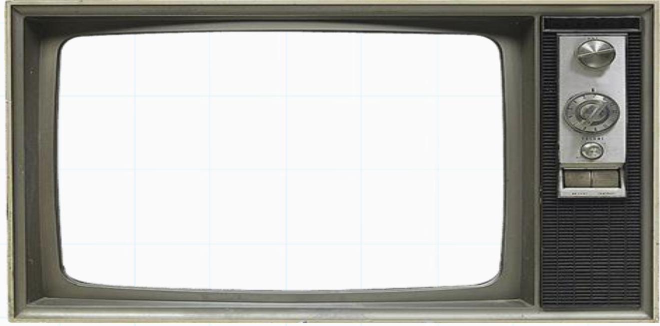
$$\sum_{j\in J}\sum_{t'\in T_{ijt}} x_{ijt'} \le 1, \quad \forall i \in M, \forall t \in H, \quad \text{onde } T_{ijt} = \{t - p_{ij} + 1, ..., t\}$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa



Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem, ela tem que ser executado num tempo superior a t mais seu tempo de execução

Formulações Clássicas

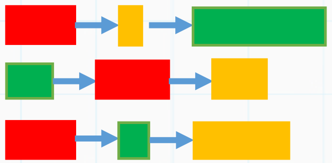


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

$$\sum_{j \in J} \sum_{t' \in T_{ijt}} x_{ijt'} \leq 1, \quad \forall i \in M, \forall t \in H, \quad \text{onde } T_{ijt} = \{t - p_{ij} + 1, ..., t\}$$

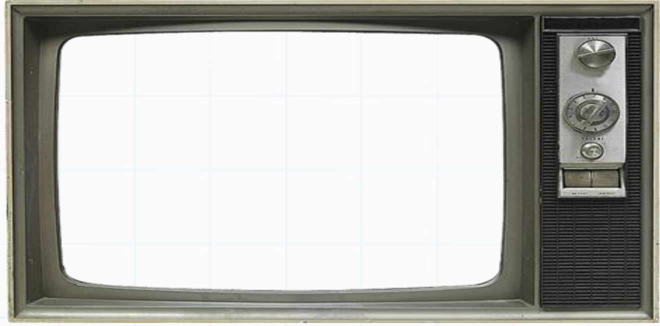
em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$\sum_{t \in H} (t + p_{\phi^j_{i-1} j}) x_{\phi^j_{i-1} jt} \qquad \forall j \in J, \forall i = 2, ..., |M|$$

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem, ela tem que ser executado num tempo superior a t mais seu tempo de execução

- Cada tarefa j ∈ J tem uma ordem de maquinas $\phi = (\phi^j_1, \phi^j_2, \ldots, \phi^j_m)$
- ϕ^j_i i-ésima máquina que vai processar tarefa j
- ϕ^j_{i-1} máquina que processou tarefa j (anterior)

Formulações Clássicas

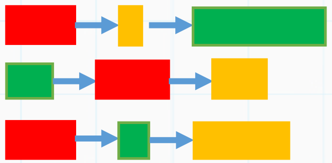


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

$$\sum_{j \in J} \sum_{t' \in T_{ijt}} x_{ijt'} \leq 1, \quad \forall i \in M, \forall t \in H, \quad \text{onde } T_{ijt} = \{t - p_{ij} + 1, ..., t\}$$

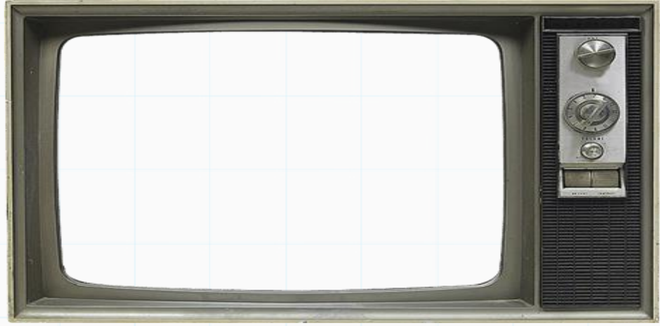
em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$\sum_{t \in H} (t + p_{\phi^j_{i-1} j}) x_{\phi^j_{i-1} j t} \leq \sum_{t \in H} t x_{\phi^j_i j t}, \quad \forall j \in J, \forall i = 2, ..., |M|$$

- Cada tarefa j ∈ J tem uma ordem de maquinas $\phi = (\phi^j_1, \phi^j_2, \dots, \phi^j_m)$
- ϕ^j_i i-ésima máquina que vai processar tarefa j
- ϕ^j_{i-1} máquina que processou tarefa j (anterior)

Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem, ela tem que ser executado num tempo superior a t mais seu tempo de execução

Formulações Clássicas

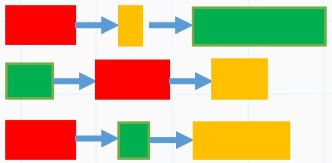


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t\in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



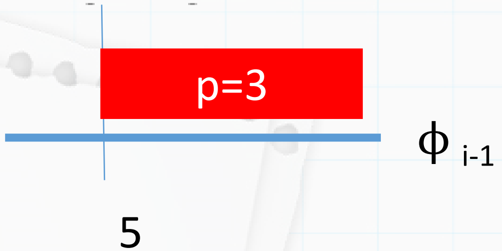
$$\sum_{t\in H}(t+p_{ij})x_{ijt} \le c_{max}, \quad \forall j\in J,\forall i\in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

$$\sum_{j\in J}\sum_{t'\in T_{ijt}} x_{ijt'} \le 1, \quad \forall i\in M,\forall t\in H, \quad \text{onde } T_{ijt}=\{t-p_{ij}+1,...,t\}$$

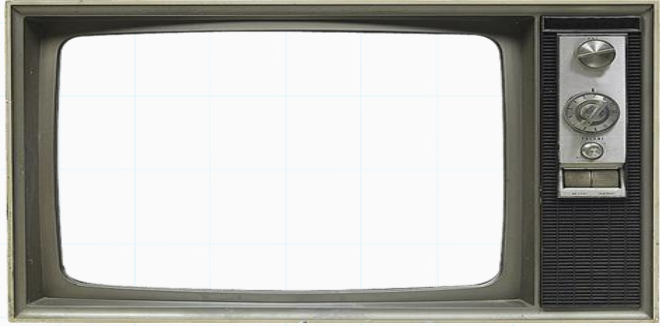
em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$\sum_{t\in H}(t+p_{\phi^j_{i-1}j})x_{\phi^j_{i-1}jt} \le \sum_{t\in H} t\, x_{\phi^j_i jt}, \quad \forall j\in J, \forall i=2,...,|M|$$



Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem, ela tem que ser executado num tempo superior a t mais seu tempo de execução

Formulações Clássicas

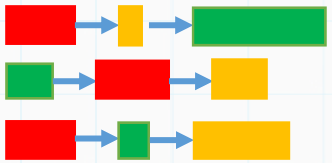


Problema do escalonamento de tarefas:

Restrições:

$$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina (x)



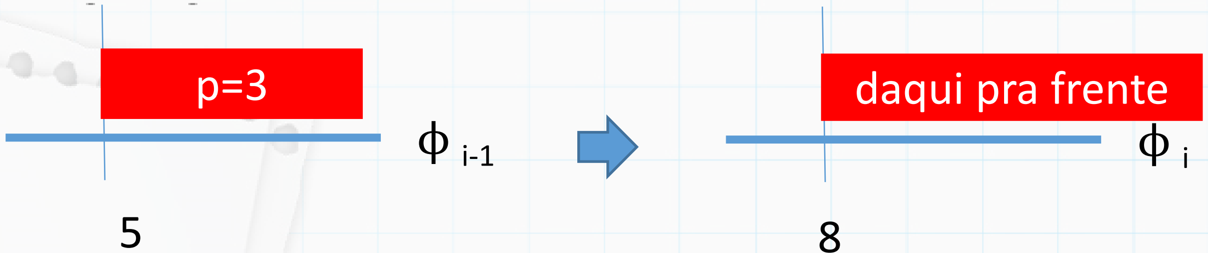
$$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa (x)

$$\sum_{j \in J} \sum_{t' \in T_{ijt}} x_{ijt'} \leq 1, \quad \forall i \in M, \forall t \in H, \quad \text{onde } T_{ijt} = \{t - p_{ij} + 1, ..., t\}$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

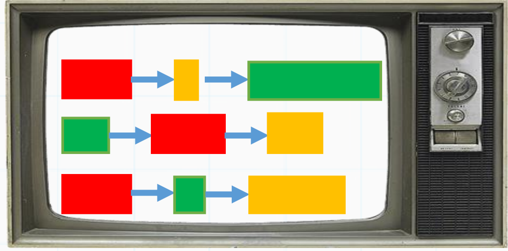
$$\sum_{t \in H} (t + p_{\phi^j_{i-1} j}) x_{\phi^j_{i-1} j t} \leq \sum_{t \in H} t x_{\phi^j_i j t}, \quad \forall j \in J, \forall i = 2, ..., |M|$$



Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem, ela tem que ser executado num tempo superior a t mais seu tempo de execução

Problema do escalonamento de tarefas:

Formulações Clássicas



$$\min \; c_{max}$$

$$\sum_{t\in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

toda tarefa tem que ser executada em toda máquina

$$\sum_{t\in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

Makespam: c_{max} tem que ser maior que o tempo final de execução de qualquer tarefa

$$\sum_{j\in J} \sum_{t'\in T_{ijt}} x_{ijt'} \leq 1, \quad \forall i \in M, \forall t \in H,$$
$$\text{onde } T_{ijt} = \{t - p_{ij} + 1, ..., t\}$$

em cada unidade de tempo na máquina, só pode estar executando uma tarefa

$$\sum_{t\in H} (t + p_{\phi^j_{i-1} j}) x_{\phi^j_{i-1} jt} \leq \sum_{t\in H} t x_{\phi^j_i jt}, \quad \forall j \in J, \forall i = 2, ..., |M|$$

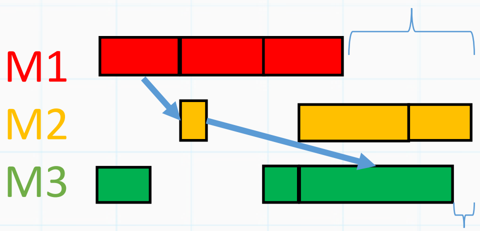
Precedência: se a tarefa foi executado num tempo t, então na sua próxima máquina na ordem, ela tem que ser executado num tempo superior a t mais seu tempo de execução

$$x_{ijt} = \begin{cases} 1, & \text{se a tarefa } j \in J \text{ começa a executar} \\ & \text{na máquina } i \in M \text{ no período } t \in T \\ 0, & \text{caso contrário} \end{cases}$$

$$c_{max} \in \mathbb{R}^+ = \{ \text{ makespam (término da última tarefa) } \}$$

<u>Problema do escalonamento de tarefas com custo financeiro:</u>

Considere o problema de escalonamento de tarefas apresentado nesta aula, mas agora cada máquina $i \in M$ possui um custo financeiro $cost_i$ por período de tempo usada (makespam), e queremos encontrar o escalonamento que minimize o custo pago pelo uso das maquinas (soma dos custos pagos).



<u>e equilibrado:</u>

Considere o problema anterior, só que agora, por uma questão operacional , os tempos de desligamento das máquinas (makespam) não podem ser muito distantes. Logo, a diferença de tempo do makespam entre quaisquer duas máquinas tem que ser no máximo E.

Modelo original

$$\min \; c_{max}$$

(1) $$\sum_{t \in H} x_{ijt} = 1, \quad \forall j \in J, \forall i \in M$$

(2) $$\sum_{t \in H} (t + p_{ij}) x_{ijt} \leq c_{max}, \quad \forall j \in J, \forall i \in M$$

(3) $$\sum_{j \in J} \sum_{t' \in T_{ijt}} x_{ijt'} \leq 1, \quad \forall i \in M, \forall t \in H,$$

(4) $$\sum_{t \in H} (t + p_{\phi^j_{i-1} j}) x_{\phi^j_{i-1} jt}$$

<u>Lembrando</u>:

- Cada tarefa $j \in J$ tem uma ordem $\phi = (\phi^j_1, \phi^j_2, \dots, \phi^j_m)$ de processamento nas máquinas
- Cada tarefa $j \in J$ e máquina $i \in M$ possui um tempo de processamento p_{ij}

1) Vars: substituir c_{max} por c_{max}^i (agora cada maquina tem um makespam)
2) F.O.: minimizar a soma dos custos, isto é, makespam de cada máquina x custo máquina.
3) Rest:
   - Adaptar restrição (2) para as novas variáveis c_{max}^i (pois precisamos de L.I. (limites inferiores) para cada makespam de cada máquina.
   - Restrições de Equilíbrio : para cada par de máquinas, a diferença de tempo final de processamento (c_{max}^i) não pode ultrapassar E

Exercício

(ESCREVA APENAS A NOVA F.O. E AS NOVAS RESTRIÇÕES)

Até a próxima