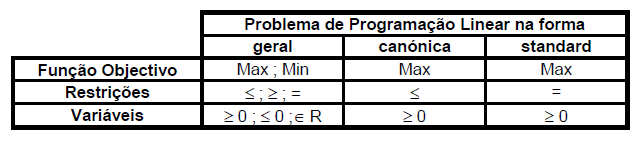
**Introdução à Investigação Operacional**

# Programação linear

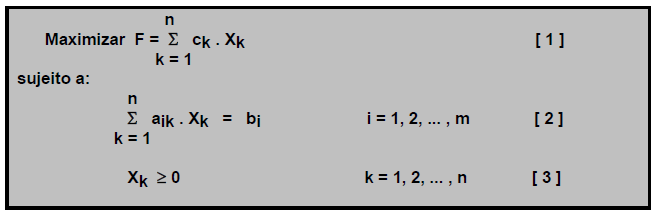


1. **Função objectivo**

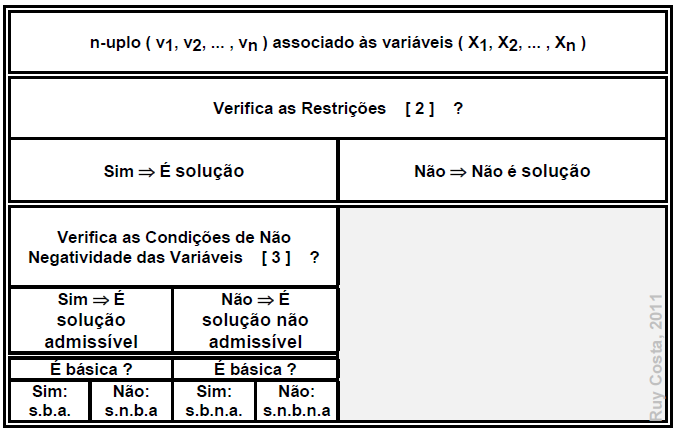
* Se Min F ⇔ Max G (com G = - F ).

1. **Restrições**
   * Canónica:
     1. Se **≥** então multiplicamos tudo por -1 para passar a **≤**
   * Standard:
     1. Se **≥** então adicionamos uma Var. de folga negativa à restição, **-F**
     2. Se **≤** então adicionamos uma Var. de folga positiva à restição, **F**
2. **Variáveis**
   * Se (Var. livre): X = X’ - X’’ tal que X’, X’’ **≥** 0
   * Se X **≥** 0então X passa a -Y

Na forma standard um um problema PL:



Relativamente a um problema PL expresso na forma standard, com **n** variáveis maior que **m** restrições (**n > m**), podem apresentar-se as seguintes definições:



Nota: solução básica 🡪 vértice

# Formulação de problemas de programação linear

1. Qual o **objectivo** a atingir?

* ... Max / Min ... → função objectivo

1. Que **actividades** devem ser levadas a cabo que influenciam o objetivo? Que **decisões** devem ser tomadas?

* ... → variáveis (intensidade)

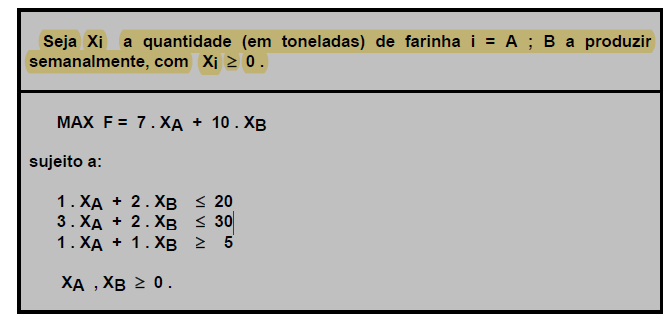
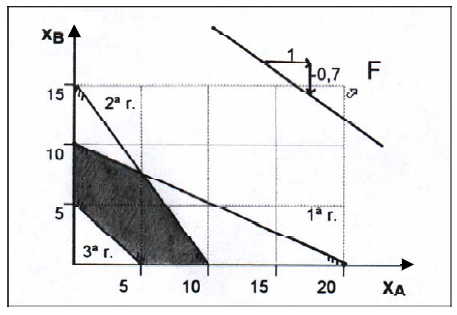
Nota:

* + Não esquecer as unidades
  + Não esquecer o carácter de não negatividade das variáveis

1. Que **recursos** são consumidos nas atividades? Que **condicionalismos** são impostos?

* ... → restrições

### Método gráfico

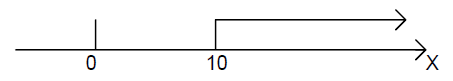
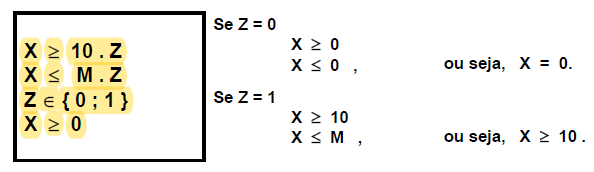
 

XB = F/10 - 7/10 . XA 🡺 declive de - 0,7 (escolher dois pontos e desenhar a reta)

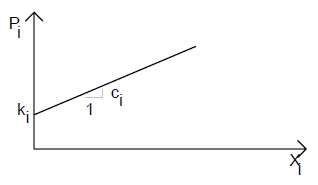
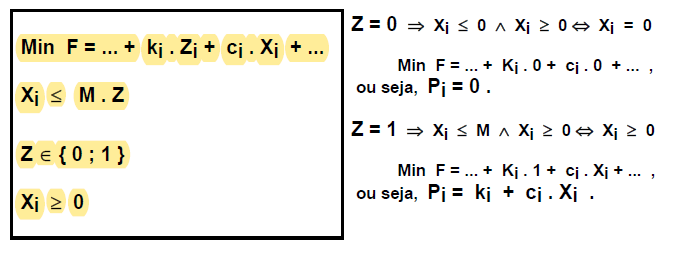
Nota: solução óptima de um problema PL é um vértice do espaço de soluções admissíveis

# Utilização de variáveis binárias na formulação de problemas PL Mista

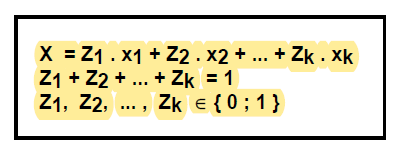
## 1 - O Lote mínimo

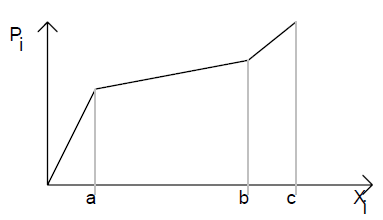
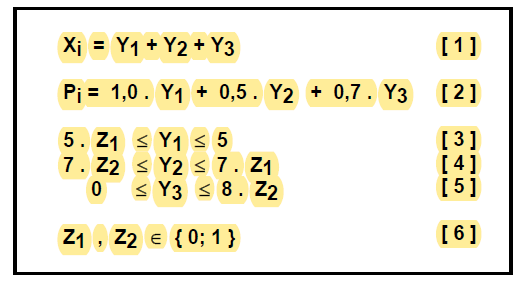
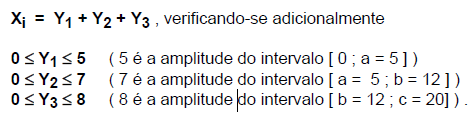
## 2 - Custo fixo de arranque de produção

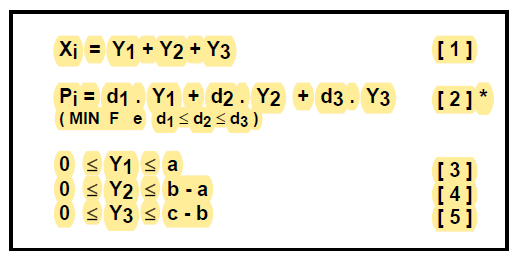
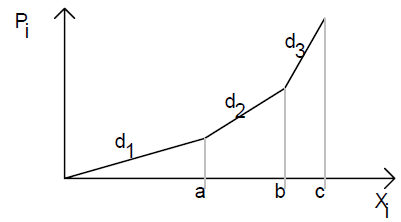
## 3 - Variável que toma valores de um dado conjunto discreto



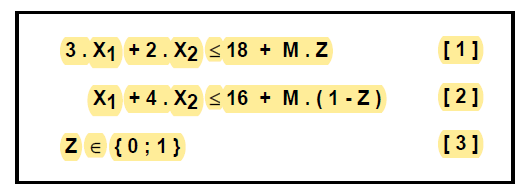
## 4 - Função objectivo com troços lineares de diferentes inclinações

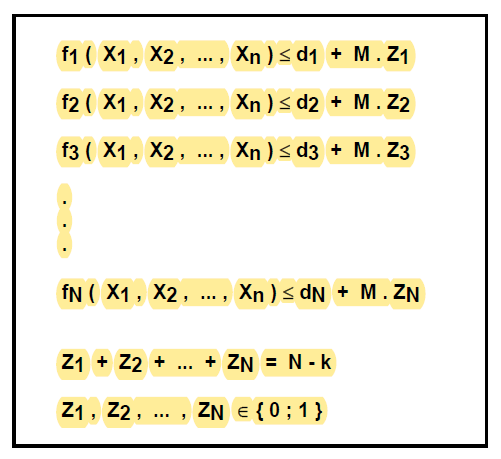
**Deseconomias de escala (minimizar com troços com declives crescentes):**



## 5 - Activação de uma de entre duas restrições

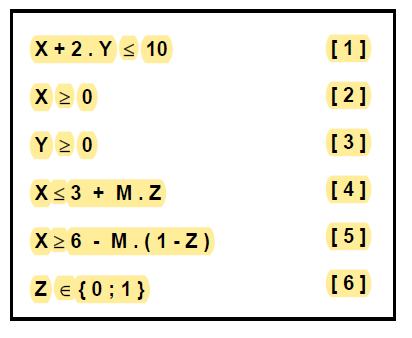
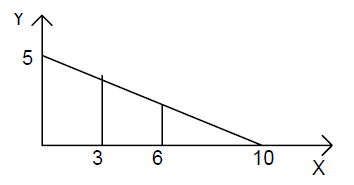


## 6 - Activação de k restrições de entre um grupo de restrições

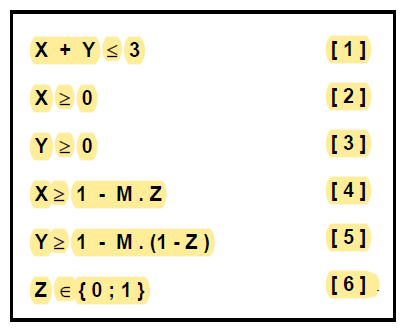
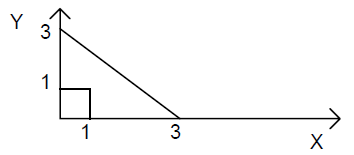


Pretende-se garantir que, de entre um grupo com mais de k restrições, sejam cumpridas, pelo menos k restrições

## 7 - Representação de domínios planos não convexos por disjunção de restrições lineares

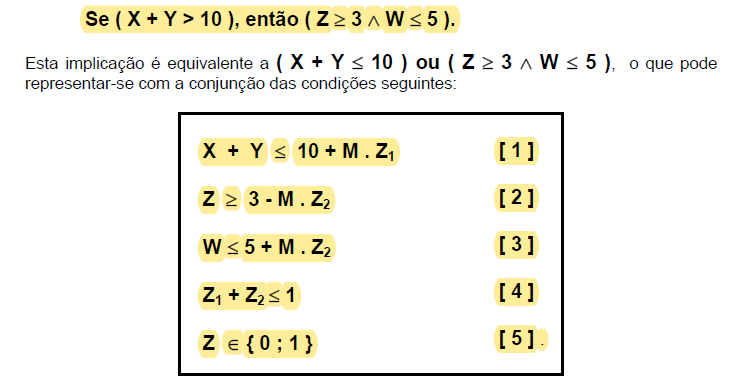


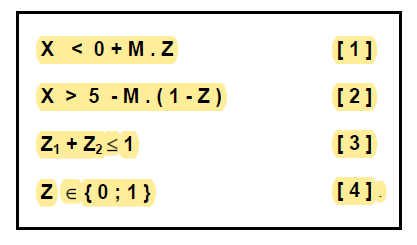
**{X + 2 . Y** ≤ **10** ∧ **X** ≥ **0** ∧ **Y** ≥ **0** ∧ **( X** ≤ **3** ∨ **X** ≥ **6 )}**



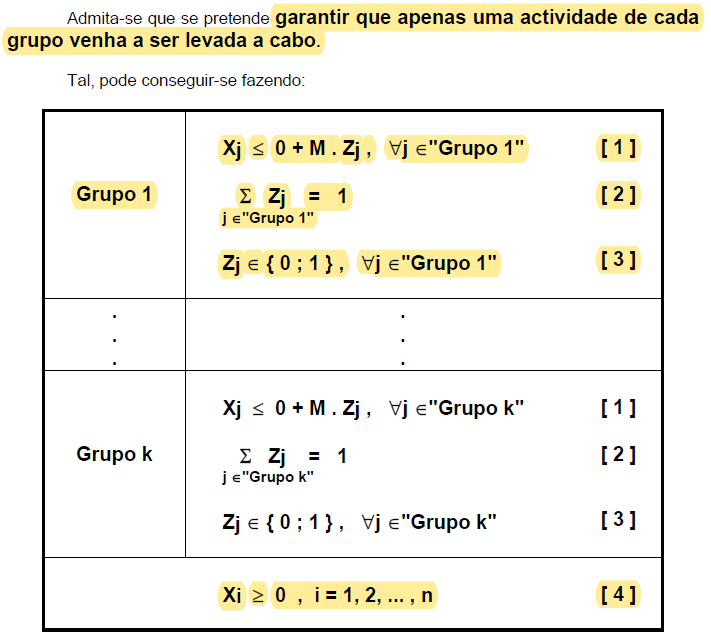
**{X + Y** ≤ **3** ∧ **X** ≥ **0** ∧ **Y** ≥ **0** ∧ **( X** ≥ **1** ∨ **Y** ≤ **1 )}**

## 8 - Implicação de restrições

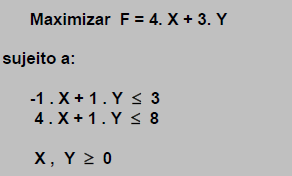


 🡺 

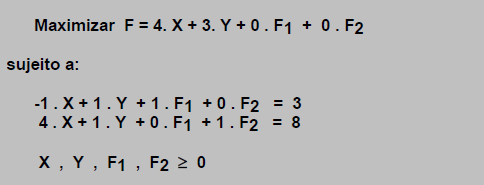
## 9 - Problemas de escolha múltipla



# Algoritmo Simplex Primal



### Re-escrever o problema na forma standard

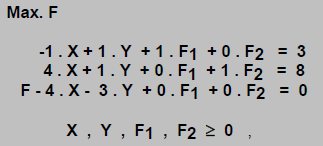


* F = 4X + 3Y + F1 + F2 🡪 F - 4X - 3Y + 0F1 + 0F2 = 0

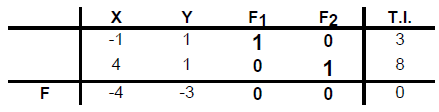
### Arbitrar uma solução básica inicial

**Regra usual:** Sempre que possível, toma-se as variáveis de folga como variáveis básicas iniciais (0,0).

Para:



Preenchemos a tablela:



**REPETIR 3, 4, 5** (até termos todos os valores positivos na última linha)

### Verificação da optimalidade da solução em análise

### Selecção da variável que entra na base

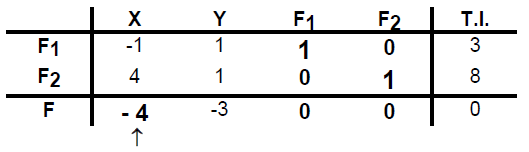
### Selecção da variável que sai da base

#### Exemplo continuação:

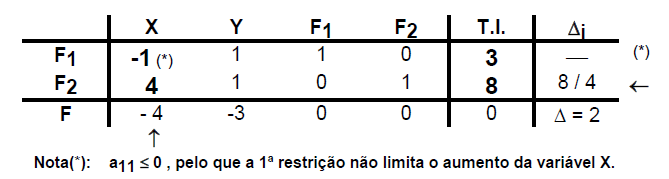
* Iteração nº1 não verifica otimalidade 🡪 to step **4**
* Iteração nº2 não verifica otimalidade 🡪 to step **4**
* Iteração nº3 verifica otimalidade 🡪 STOP

Nota: Deve ser incrementada a variável com o coeficiente mais negativo na linha correspondente à

função objectivo

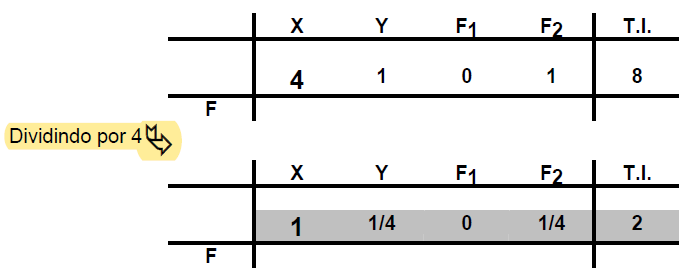


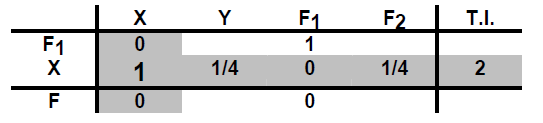
* Iteração nº1:



Preencher linha pivo

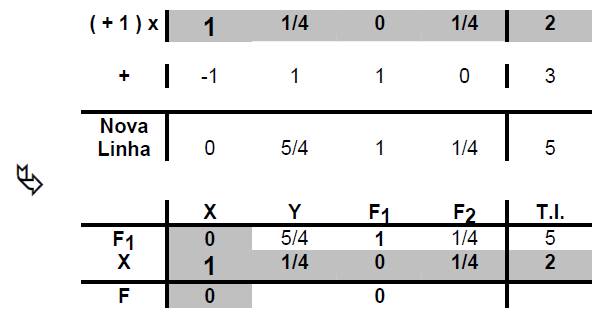
**Perguntar qual o nº que a multiplicar por 4 dá 1 ( ¼ neste caso)**



**Ficamos com:** 

* Simétrico do coeficiente do Quadro anterior correspondente à variável que entrou para a base:

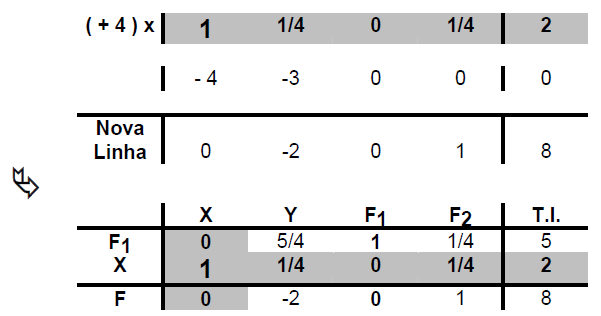
**- ( -1) = + 1 - - - - - >** PARA **F1**



**Para atualizar a linha F1 temos de ver o antigo valor na coluna pivot, e multiplicar a linha pivot pelo simétrico desse valor, depois é só somar essa linha com a antiga F1**

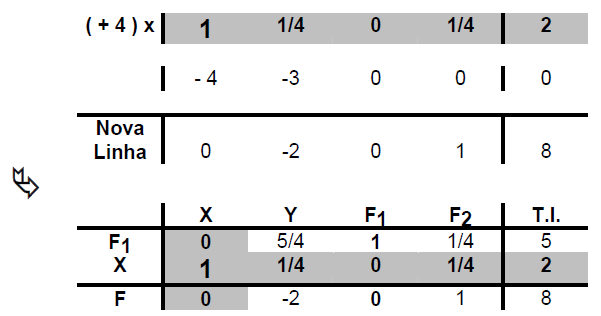
* Simétrico do coeficiente do Quadro anterior correspondente à variável que entrou para a base:

**- ( -4) = + 4 - - - - - >** PARA **F**

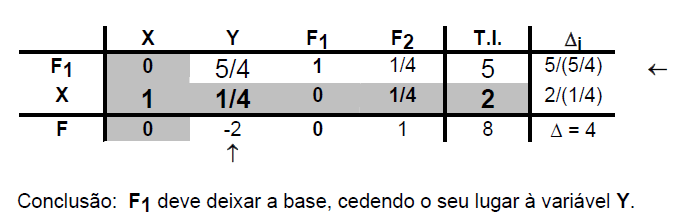


**Para atualizar a linha F temos de ver o antigo valor na coluna pivot, e multiplicar a linha pivot pelo simétrico desse valor, depois é só somar essa linha com a antiga F**

* Colocando tudo na tabela temos a preenchida 😊



A partir daqui é repetir



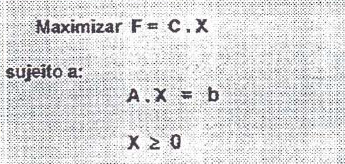
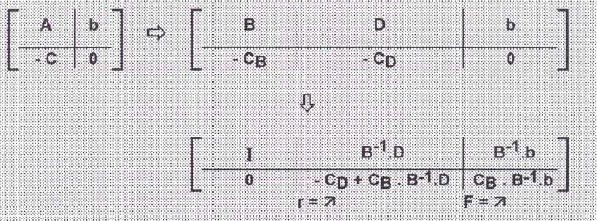
**. . .**

## Importante

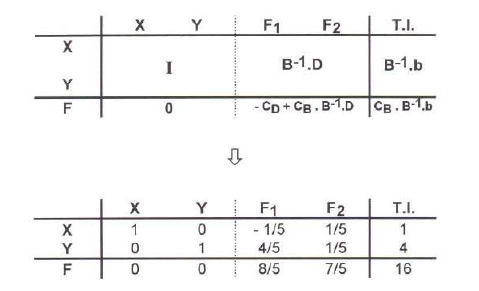
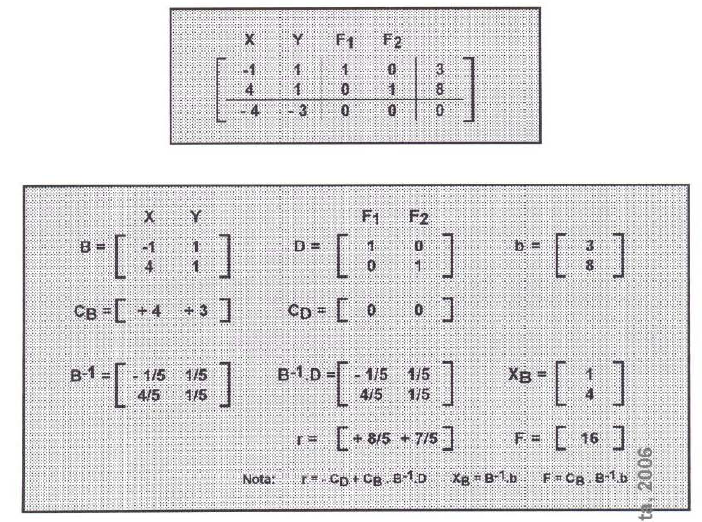
Caso os deltas sejam infinito 🡺 ou seja a divisão do calculo é negativo, podemos ter:

* Os valores na linha F todos >= 0, e neste caso: Existe mais do que uma solução ótima
* Exista valores na linha F < 0, então: Espaço de soluções admissíveis ilimitado e sem uma solução óptima

# Formulação matricial do simplex

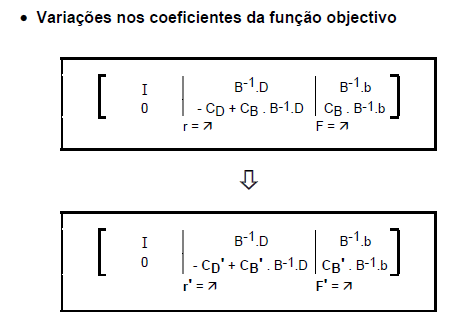
## Exemplo:



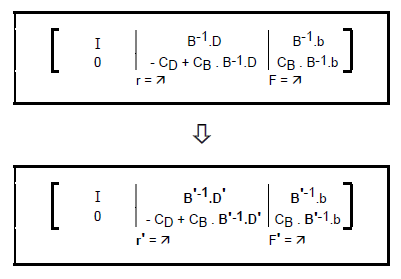
**Importante:**

B^-1 \* D = o valor é um inverso de uma coluna de B^-1

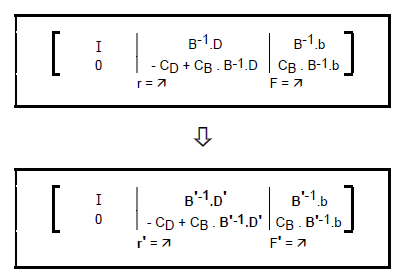
**Variação nos coeficientes da função objetivo**



**Variação nos termos independentes das restrições**



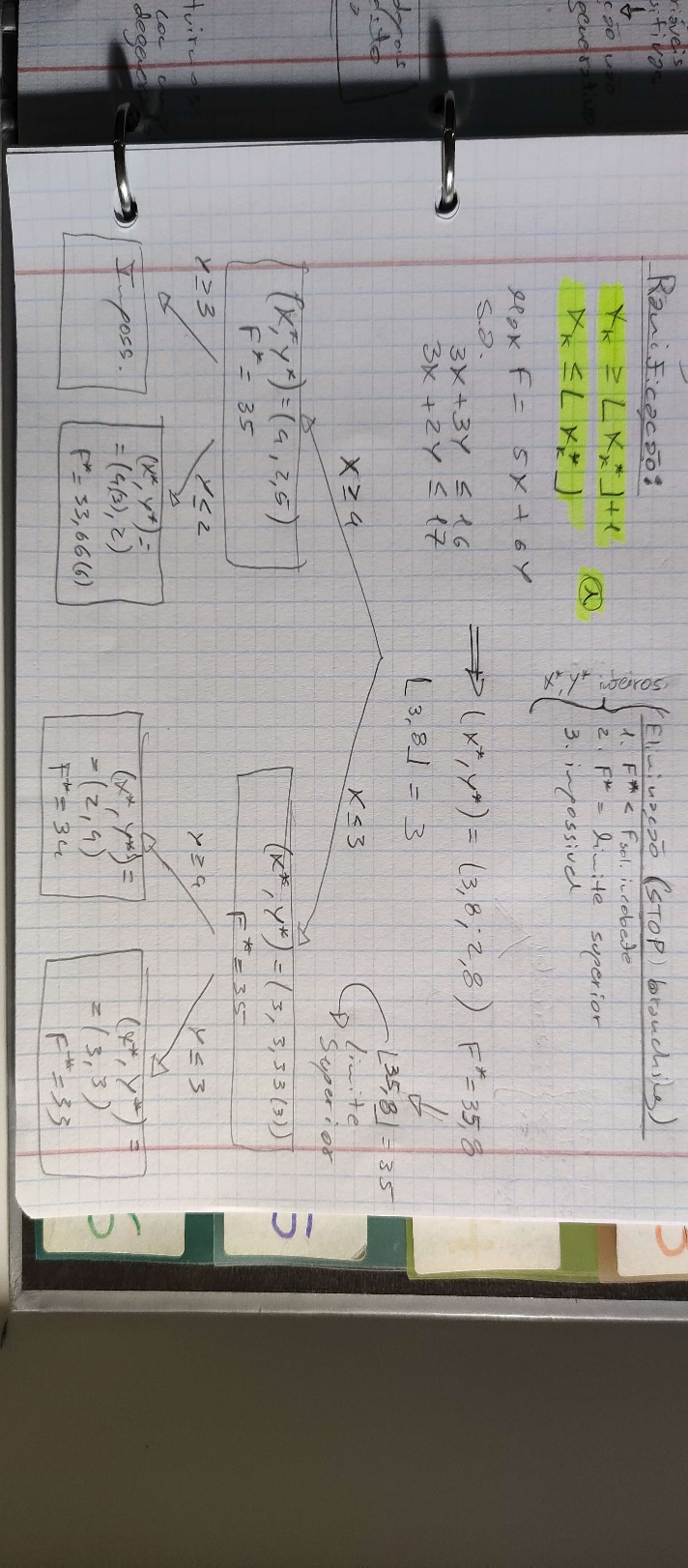
**Variação nos coeficientes das restrições**



**Introdução de novas variáveis**

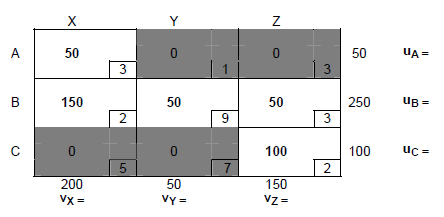
* rn = -Cn + CB \* B^-1\*dn

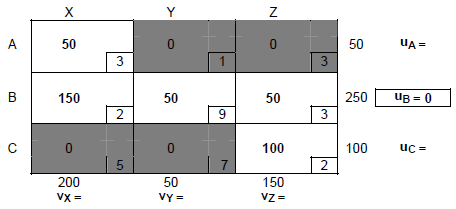
# Algoritmo Branch and Bound

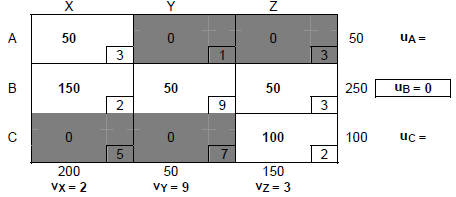


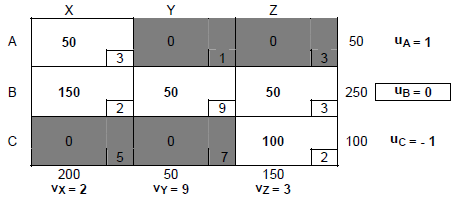
# Algoritmo dos transportes

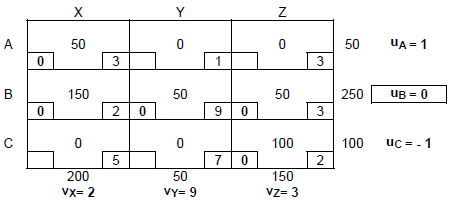
Cij’ = Cij - ui - vj 🡪 para var. básicas (Vy = 0 ) temos que: Cij’ = 0 ʌ Cij = ui - vj

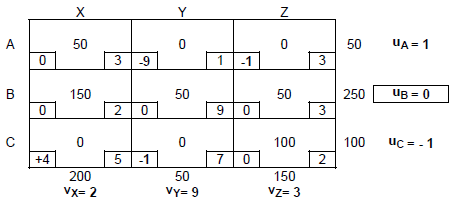


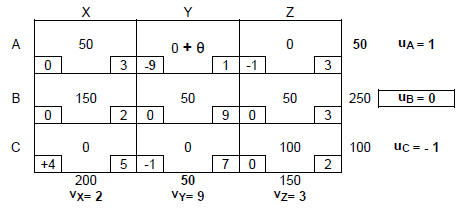
 **escolher** **linha com mais var. básicas**

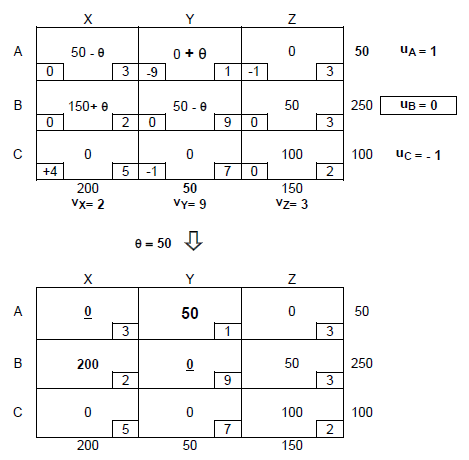
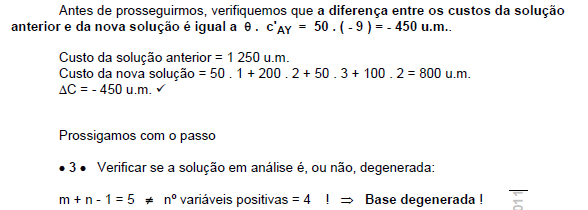






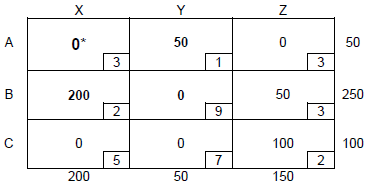
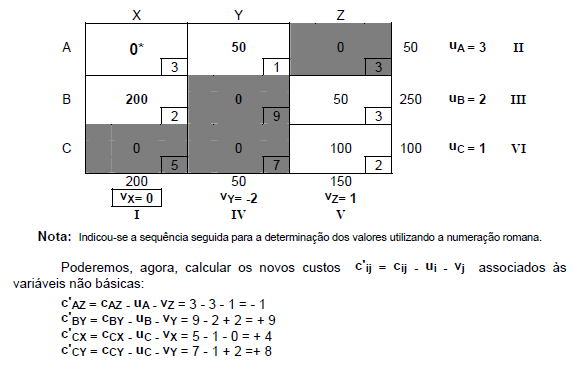


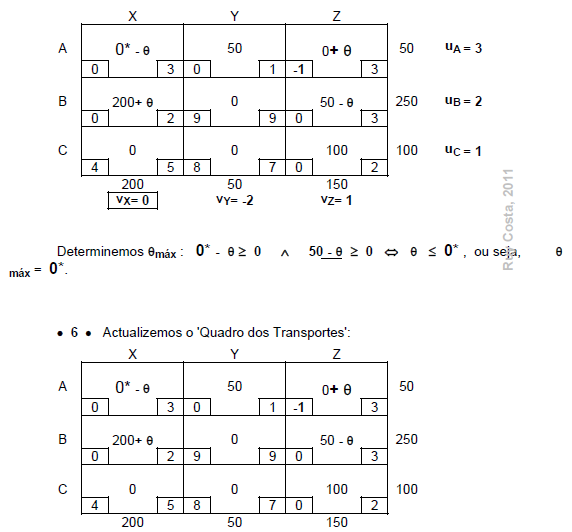
 **maior cij’**

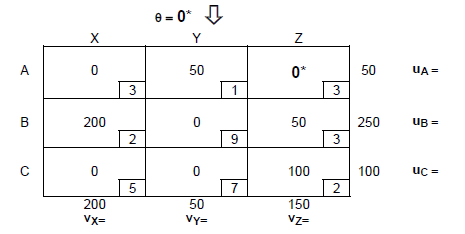
 

Quando mais do que uma variável deixa a base, é conveniente assinalar (por exemplo, sublinhando) essas variáveis, para posteriormente se escolher a(s) variável(eis) a 'promover' artificialmente para a base.

Como regra geral, e uma vez que se pretende minimizar a função objectivo, escolheremos para integrar artificialmente na base a variável de menor custo, de entre as candidatas

 **🡪** 





**Voltamos ao mesmo . . .**

