Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Engenharia de Produção

# Tópicos Especiais em Pesquisa Operacional
## (Período 2020.3)

## 5ª Atividade Assíncrona
## (Problema de Programação da Produção – *Scheduling*)

1. Ler os capítulos/seções do livro "*Pesquisa Operacional para cursos de engenharia*" (de Arenales et al., 2015) abaixo mencionados e elaborar um resumo de no máximo uma página. (15%)
   a) Capítulo 3 (seções 3.6.1, 3.6.2, 3.6.3 e 3.6.4).

2. Considerando a instância abaixo, responda as seguintes questões:
   a) Encontre uma solução ótima para um problema de Scheduling do tipo (P | | C max ) resolvendo o modelo (1)-(5) apresentado nos slides da aula. Qual o valor da solução ótima? Qual a solução encontrada?
   b) Encontre uma solução ótima para um problema de Scheduling do tipo (P | | C max ) utilizando o algoritmo de busca linear. Qual o valor da solução ótima? Qual a solução encontrada? Quantas iterações foram necessárias?
   c) Encontre uma solução ótima para um problema de Scheduling do tipo (P | | C max ) utilizando o algoritmo de busca binária. Qual o valor da solução ótima? Qual a solução encontrada? Quantas iterações foram necessárias?
   d) Para cada uma das soluções encontradas, crie um gráfico de Gantt (similar ao apresentado no Slide 4 da aula) para representar a solução obtida

**Instância:**
- **Número de máquinas: m = 3,**
- **Número de tarefas: n = 21, e**
- **Tempos de processamento das tarefas: p = [2 1 9 7 5 7 3 4 5 8 6 4 3 2 2 1 4 3 5 5 3]**

**Enviar os seguintes arquivos:**

1. Arquivo .pdf contendo o resumo dos capítulos e/ou seções mencionados e as respostas das questões.
   Nomear o arquivo da seguinte maneira: Atividade5-NOME-SOBRENOME.pdf

2. Os códigos (arquivos .jl) implementados.
   Nomear os arquivos da seguinte maneira: Atividade5-codigo1-NOME-SOBRENOME.jl, Atividade5-codigo2-NOME-SOBRENOME.jl, etc.

**Referência:**

**ARENALES, M.; ARMENTANO, V.; YANASSE, H.; MORABITO, R. Pesquisa operacional para cursos de engenharia. 2. ed. Elsevier, 2015.**

**OBS**: Alguns comandos básicos de Julia/JuMP são apresentados na página seguinte

UFPE
**Universidade Federal de Pernambuco**
**Centro de Tecnologia e Geociências**
**Departamento de Engenharia de Produção**

# Comandos básicos de Julia+JuMP

**- Como declarar um vetor em Julia:**

- myvec = Int64[] # Declara um vetor de valores inteiros
- myvec = Floar64[] # Declara um vetor de valores reais
- myvec = Bool[] # Declara um vetor de valores binários (booleanos)

**- Para inserir elementos (ou valores) em um vetor:**

- push!(myvec, elem) # Insere o elemento "elem" no final do vetor "myvec"
- push!(myvec, 2) # Insere o valor 2 no final do vetor "myvec"
- pushfirst!(myvec, 10) # Insere o valor 10 no início do vetor "myvec"
- insert!(myvec, pos, elem) #Insere o elemento "elem" no vetor "myvec" antes do elmento de posição "pos"
  - Por exemplo: Seja myvec = [5, 10, 15, 22, 28]. O comando insert!(myvec, 3, 999) irá inserir o elemento 999 antes do 3° elemento (15), resultando em: myvec = [5, 10, 999 15, 22, 28]

**- Para remover elementos de um vetor:**

- pop!(myvec) # remove o último elemento do vetor "myvec"
- popfirst!(myvec) # remove o primeiro elemento do vetor "myvec"
- deleteat!(myvec, pos) # remove o elemento da posição "pos"
- unique!(myvec) #remove os elementos repetidos de um vetor

**- Para ordenar um vetor:**

- sort!(myvec) # ordena o vetor "myvec" em ordem crescente
- sort!(myvec, rev=true) # ordena o vetor "myvec" em ordem decrescente

**- Para arredondar um valor real ("v") para um valor inteiro:**

- x = floor(Int64, v) # arredonda o valor de v para o inteiro inferior e armazena em x
  - Por exemplo: x = floor(Int64, 5.75) irá salvar o valor 5 em x.

**- Como declarar uma matriz bi-dimensional:**

modo1) matriz = Array{Int64}(undef, 2, 4) # Declara uma matrix 2x4 de valores inteiros
modo2) matriz = zeros(Bool, 2, 3) # Declara uma matrix 2x3 de valores binários, inicializados com zeros
modo3) matriz = [1 4 3 1; 1 1 1 2] # Declara uma matrix 2x4, atribuindo os valores manualmente
Exemplos de outros tipos de variáveis: Float64 (real); Bool (binário); Char (caractere); Int64 (inteiro)
Mais detalhes em: https://docs.julialang.org/en/v1/manual/types/

**- P/ acessar um elemento de uma matriz bi-dimensional:**

valor = matriz[1, 2] # acessa o elemento (valor) da (linha 1, coluna 2).

**- Como declarar uma restrição de igualdade em Julia+JuMP**

Para uma restrição do tipo: x + y = 100, fazemos: `@constraint(model, x + y == 100)`

**- P/ declarar um conj. de variáveis binárias com 2 índices (ex: x11, x12, …, x19; x21, x22, …, x29; (…) x91, x92, …, x99)**

`@variable(model, x[i=1:9, j=1:9], Bin)`. Após resolver o modelo, o valor da variável xij podem ser obtido assim: sol[i,j] = value.(x[i,j])

OBS: Atenção para distinguir uma variável de Julia com uma variável do modelo matemático. Variáveis do modelo matemático são declaradas com `@variable(param1, param2, param3)`, onde param1 se refere ao nome modelo, param2 se refere ao nome da variavel e param3 ao tipo da variável.