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

Tópicos Especiais em Pesquisa Operacional
(Período 2020.3)

3ª Atividade Assíncrona
(Problemas de Localização)

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.
   a) Capítulo 3 (seção 3.5.2);

2. Implemente o modelo matemático do problema dos p-centros, resolva a instância em anexo (instanciaPCP.txt), para p = 2, 3 e 5, e responda:
   a) Para p = 2, qual a solução ótima? Qual o valor da solução ótima? Qual o tempo de execução?
   b) Para p = 3, qual a solução ótima? Qual o valor da solução ótima? Qual o tempo de execução?
   c) Para p = 5, qual a solução ótima? Qual o valor da solução ótima? Qual o tempo de execução?

3. Utilizando o algoritmo de **busca linear** apresentado na aula do dia 23/09, resolva a instância instanciaPCP.txt (em anexo) do problema dos p-centros, também para p = 2, 3 e 5, e responda (para cada valor de p):
   a) Qual o valor da solução ótima?
   b) Quantas iterações foram necessárias (isto é, quantos problemas de cobertura foram resolvidos)?
   c) Qual o tempo de execução do algoritmo?

4. Utilizando o algoritmo de **busca binária** apresentado na aula do dia 23/09, resolva a instância instanciaPCP.txt (em anexo) do problema dos p-centros, também para p = 2, 3 e 5, e responda (para cada valor de p):
   a) Qual o valor da solução ótima?
   b) Quantas iterações foram necessárias (isto é, quantos problemas de cobertura foram resolvidos)?
   c) Qual o tempo de execução do algoritmo?

5. Com base nas respotas obtidas nas questões 2, 3 e 4, qual método você indicaria pra resolver o problema dos p-centros? Por que?

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: Atividade3-NOME-SOBRENOME.pdf

2. Os códigos (arquivos .jl) implementados.
   Nomear os arquivos da seguinte maneira: Atividade3-codigo1-NOME-SOBRENOME.jl, Atividade3-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

Destaquei em amarelo alguns comandos necessários para a realização da atividade.

- 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 3º 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.