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

Tópicos Especiais em Pesquisa Operacional
(Período 2020.3)

4ª Atividade Assíncrona
(Problema do Caixeiro Viajante)

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 (seção 3.4.5);

2. Pesquise sobre o Problema do Caixeiro Viajante (PCV) e responda as seguintes questões:
   a) Como pode ser definido o PCV? (10%)
   b) Qual a diferença entre o PCV simétrico e o PCV assimétrico? (5%)
   c) Apresente a formulação MTZ proposta por Miller, Tucker and Zemlin (1960) para o PCV e explique o significado de cada restrição. (15%)
   d) Qual a vantagem da formulação MTZ em relação à formulação clássica de eliminação de subciclos (e.g., apresentada em Arenales et al., 2015)? (15%)
   e) Cite três aplicações práticas do Problema do Caixeiro Viajante. (7.5%)
   f) Além das variantes mencionadas no livro de Arenales et al. 2015 (m-caixeiros viajantes; caixeiro-viajante aquisição; e caixeiro-viajante lucro), cite outras duas variantes do PCV e explique como são definidas. (7.5%)
   g) Qual a solução ótima para a instância do PCV abaixo: (25%)

Número de cidades (n): 10

Coordenadas:

cidade	x	y
1	651.19	2244.39
2	676.60	2160.20
3	701.99	2162.20
4	727.40	2165.40
5	752.79	2168.60
6	727.40	2132.09
7	701.99	2142.19
8	676.60	2152.39
9	651.19	2162.59
10	625.80	2172.79

Matriz de distâncias:

	1	2	3	4	5	6	7	8	9	10
1	0	88	97	110	127	136	114	95	82	76
2	88	0	25	51	77	58	31	8	26	52
3	97	25	0	26	51	39	20	27	51	77
4	110	51	26	0	26	33	34	52	76	102
5	127	77	51	26	0	44	57	78	102	127
6	136	58	39	33	44	0	27	55	82	109
7	114	31	20	34	57	27	0	27	55	82
8	95	8	27	52	78	55	27	0	27	55
9	82	26	51	76	102	82	55	27	0	27
10	76	52	77	102	127	109	82	55	27	0

OBS: Para simplificar o processo de resolução dos problemas, ao calcular a matriz de distâncias, arredondem os valores calculados para o inteiro inferior. Por exemplo, para todo i, j:

```
dx = coords[i,1]-coords[j,1]
dy = coords[i,2]-coords[j,2]
d[i,j] = floor(Int64, sqrt(dx * dx + dy * dy))
```

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: Atividade4-NOME-SOBRENOME.pdf

2. Os códigos (arquivos .jl) implementados.
   Nomear os arquivos da seguinte maneira: Atividade4-codigo1-NOME-SOBRENOME.jl, Atividade4-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 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.