Estudo Caso 1: Otimização

Mário Leite

...

Otimizar um determinado processo é encontrar valores que fazem o (s) resultado (s) se tornarem os melhores possíveis, numa determinada situação em que certas restrições devam ser levadas em consideração para que essa otimização seja possível; em outras palavras, "otimizar" é extrair o melhor rendimento de um processo, atendendo a determinadas condições". Por exemplo, numa instituição bancária em que esteja havendo reclamando devido ao pequeno número de caixas para atender os cientes: "*Qual seria o número de caixas necessários para tender todos os clientes, sem, no entanto, aumentar muito as despesas do banco?! Qual seria o número ÓTIMO de funcionários extras para tender as reclamações dos clientes?! Em qual horário TODOS os caixas deveriam ser ativados?!* Em situações semelhantes a esta é que se faz necessário a OTIMIZAÇÃO do processo...

Estudo de Caso: ***"Uma companhia de navegação possui três tipos de recipientes A, B e C, que transporta cargas em containers de três tipos: I, II e III".***

As capacidades dos recipientes são dadas no ***quadro 1***.

Recipiente	I	II	III
A	4	3	2
B	5	2	3
C	2	2	3

Quadro 1 - Recipientes x Contêiners

Questão a ser resolvida: "*Quais devem ser os números de recipientes* ***x1***, ***x2*** *e* ***x3*** *de cada categoria* (A, B e C), *se a companhia deve transportar* ***42*** *containers do tipo* ***I, 27*** *do* ***tipo II*** e ***33*** *do* ***tipo III***?"

Montagem do sistema de equações lineares

$$\begin{cases} 4x1 + 5x2 + 2x3 = 42 \\ 3x1 + 2x2 + 2x3 = 27 \\ 2x1 + 3x2 + 3x3 = 33 \end{cases}$$

O programa **"ProblemaOtimizacao1"**, codificado em Python, resolve facilmente esse problema dando os valores ótimos dos tipos de recipientes para fazer o transporte da melhor maneira possível. A **figura 1** mostra a saída do programa com os resultados da Otimização

---

Nota: As variáveis começando com as iniciais "**Lst**" no programa é uma técnica de programação do próprio autor para indicar que se trata de elementos estruturais indexados; no caso: "*listas*". Mas, poderia ser qualquer identificador dentro das regras de nomeação de variáveis da linguagem implementada.

```
'''
ProblemaOtimizacao1.py
--------------------------------------------------------------------------------
Resolve o problema de otimização dos tipos de recipientes da empresa
transportadora de contêiners.
--------------------------------------------------------------------------------
Autor: Mário Leite
Data: 18/02/2024
Ambiente: PyCharm
--------------------------------------------------------------------------------
'''

from itertools import product

class ClsConteiner:
    def __init__(self, LstCapacs):
        self.LstCapacs = LstCapacs

    def ResolverProblema(self, LstDemandas):
        melhorComb = None
        menorNumRecips = float('inf')

        for quantA in range(0, LstDemandas[0]+1):
            for quantB in range(0, LstDemandas[1]+1):
                for quantC in range(0, LstDemandas[2]+1):
                    #Verifica se a combinação atual satisfaz as demandas e
                  #utiliza o menor número de recipientes
                    if(self.VerificarSatisfacao(quantA, quantB,
                    quantC, LstDemandas)):
                        numRecips = quantA + quantB + quantC


                        if(numRecips < menorNumRecips):
                            melhorComb = (quantA, quantB, quantC)
                            menorNumRecips = numRecips

        #Mostra os resultados
        if(melhorComb):
            print("Melhor combinação de recipientes:")
            print("Número de recipientes A:", melhorComb[0])
            print("Número de recipientes B:", melhorComb[1])
            print("Número de recipientes C:", melhorComb[2])
            print("Número total de recipientes:", menorNumRecips)
        else:
            print("Não foi possível encontrar uma solução satisfatória!")
#-------------------------------------------------------------------------------
    def VerificarSatisfacao(self, quantA, quantB, quantC,
        LstDemandas):
        capacA = self.LstCapacs[0][0] * quantA + self.LstCapacs[1][0]
        * quantB + self.LstCapacs[2][0] * quantC
        capacB = self.LstCapacs[0][1] * quantA + self.LstCapacs[1][1]
        * quantB + self.LstCapacs[2][1] * quantC
        capacC = self.LstCapacs[0][2] * quantA + self.LstCapacs[1][2]
        * quantB + self.LstCapacs[2][2] * quantC

        return capacA >= LstDemandas[0] and capacB >= LstDemandas[1]
               and capacC >= LstDemandas[2]
```

```
#=========================================================================================
#Programa principal
#Capacidades dos recipientes (restrições do problema)
LstCapacs = [
    [4, 3, 2],   #recipiente A
    [5, 2, 3],   #recipiente B
    [2, 2, 3]    #recipiente C
]

#LstDemandas
LstDemandas = [42, 27, 33]  #demanda paras tipos I, II e III, respectivamente

#Criar objeto da classe "ClsConteiner" e resolver o problema
Conteiner = ClsConteiner(LstCapacs)
Conteiner.ResolverProblema(LstDemandas)
#Fim do programa "ProblemaOtimizacao1" ---------------------------------------
```

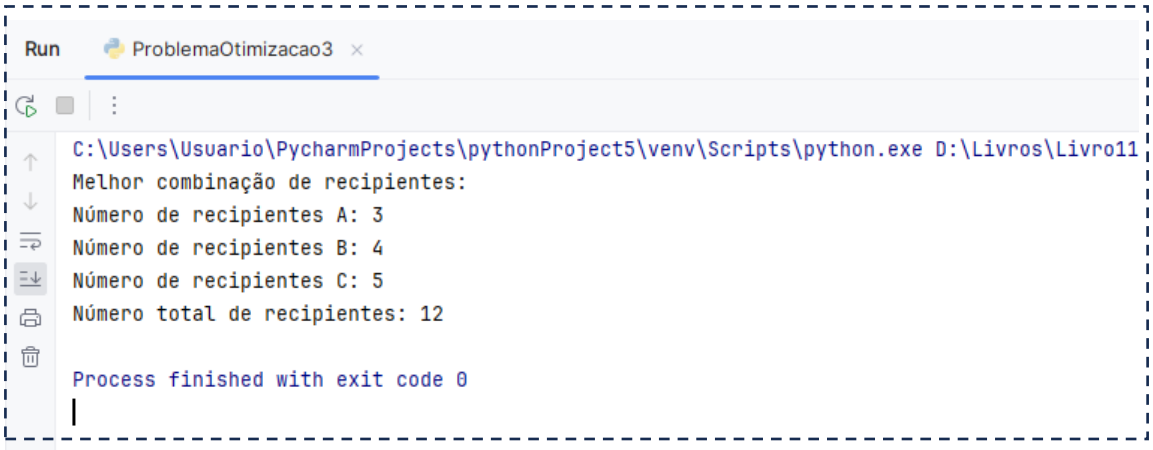

Figura 1 - Saída do programa "ProblemaOtimizacao1"