Projeto II

Planejamento de produção 📦

Dupla: - Renan Almeida Goes, 20170021539

- Lucas Moreira, 20170027336

Repositório do GitHub:

Renan-Goes/integer-programming

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together. Millions of developers and companies build, ship, and maintain their software on GitHub - the largest and most advanced development platform in the world. You can't

https://github.com/Renan-Goes/integer-programming

Introdução

Uma fábrica precisa produzir *n* produtos a partir de *m* matérias-primas. Para produzir 1 unidade de um produto *i* exige *aij* unidades de uma matéria-prima *j* e *bi* horas na linha de produção. Há, outrossim, uma disponibilidade total de *B* horas da linha de produção durante um mês ao custo fixado de *F* reais. Para trocar de um produto na linha gastam-se *T* horas. Em que lotes fechados com *Lj* unidades de matéria-prima *j* e podem ser comprados por um total de *Cj* reais cada lote. os produtos **podem ou não** ser produzidos. Caso seja produzido um determinado produto *i*, ele deverá estar dentro de um intervalo de demanda mínima (*DMINi*) e uma máxima (*DMAXi*). Uma unidade de um dado produto *i* pode ser vendida por *Ri* reais. Quanto de cada produto deve ser produzido para que se maximize o lucro da fábrica.

Modelagem

Por exemplo: n = 3, m = 2, B = 450, T = 20 e F = R$50.000

	p1	p2	p3	L	C (R$)
m1	3	1	2	500	5000,00
m2	2	3	1	600	8000,00
b	0,25	0,15	0,10		
DMIN	500	600	400		
DMAX	1100	1200	800		
R (R$)	100,00	70,00	50,00		

Modelagem para o problema em questão, a posteriori, será desenhada a generalização desse problema:

n → Quantidade de produtos

m → Quantidade de matérias-primas

Para cada produto *i*, *aij* unidades são necessárias para a matéria-prima *j*

B → Horas na linha de produção por mês

F → Custo por mês na linha de produção em R$

T → Tempo necessário para se trocar um produto

Lj → Quantidade por lote de uma dada matéria-prima j

Cj → Custo por lote de uma dada matéria-prima j

Ri → Preço para um produto i

bi → Horas na linha de produção para cada produto i

ki → Produto existe ou não (variável binária)

wj → Quantidade de lotes de cada matéria-prima j

aij → Quantidade de matéria prima j utilizada pelo produto i

Dessa forma, fez-se necessário a adição de duas novas variáveis: ki e wj, que possam definir se um produto deve ou não ser produzido, conforme definido no problema e a quantidade de lotes, que é o deve ser escolhido na solução para definir quanto de matéria-prima será comprada

Dadas essas variáveis, e a explicação do problema na introdução o lucro será calculado da seguinte maneira:

$$n1 * R1 + n2 * R2 + n3 * R3 - F - (w1 * C1 + w2 * C2)$$

Isso porque o lucro gerado será dado pelos produtos vendidos, logo, a quantidade de cada produto pelo seu preço unitário para cada produto. Contudo, a produção terá um custo que deve ser subtraído do lucro, que seriam tanto o custo fixo, quanto a quantidade de lotes vendidos de cada matéria-prima pelo preço de cada lote dela

O qual está sujeito às seguintes **restrições**:

Para cada matéria-prima j, será calculada a quantidade utilizada por todos os produtos e será verificado se essas quantidades são menores ou iguais às advindas dos lotes

$a11 * n1 + a21 * n2 + a31 * n3 <= w1 * L1$

$a12 * n1 + a22 * n2 + a32 * n3 <= w2 * L2$

Restrição de tempo, delimitando que cada unidade de cada produto terá bi horas na linha de produção para cada produto i, ademais, T representa o tempo necessário para realizar a troca de um produto que está relacionado ao somatório das variáveis binárias de se um produto foi produzido ou não (ki), sendo assim, a quantidade horas gastas por trocas só serão adicionadas para produtos que foram produzidos. Tudo isso deverá ser menor ou igual a B, representando a disponibilidade de horas da linha de produção em um mês

$n1 * b1 + n2 * b2 + n3 * b3 + (\Sigma ki) * T <= B$

Não é necessário produzir um dado produto i, se este for comprado, faz-se necessário respeitar as restrições de quantidade determinadas por DMINi (quantidade mínima de um produto i) e DMAXi (quantidade máxima de um produto i), assim, fazendo o produto de DMINi e DMAXi com ki, pode-se definir se o produto será ou não comprado, já que quando ki for zero, a quantidade a ser vendida será dada pelo limite [0, 0], enquanto que quando foi um, esse limite será [DMINi, DMAXi]

$n1 >= DMIN1 * k1$

$n1 <= DMAX1 * k1$

$n2 >= DMIN2 * k2$

$n2 <= DMAX2 * k2$

$n3 >= DMIN3 * k3$

$n3 <= DMAX3 * k3$

Generalização para qualquer problema:

Para podermos generalizar as restrições, consideramos qualquer quantidade *n* de produtos e *m* de materiais, utilizando o *para todo* e o somatório para cobrir todos os produtos *i* e todos os materiais *j*

$\forall i, DMINi * ki \ <= ni <= DMAXi * ki$

$\Sigma ni * bi + (\Sigma ki) * T <= B$

$\forall j, \Sigma aij * ni <= wj * Lj$

Função Objetivo:

Sendo assim, a função objetivo do problema representando o lucro da fábrica será a seguinte. Temos que, todos os produtos produzidos serão multiplicados pelo seu preço, o que será subtraído do custo fixo mensal da linha de produção e da quantidade total de lotes de matérias-primas utilizadas associadas ao seu custo, essa função será limitada pelas restrições supracitadas:

$(\sum_{i=0}^{n} ni * Ri) - F - (\sum_{j=0}^{m} wj * Cj)$

Especificação da entrada

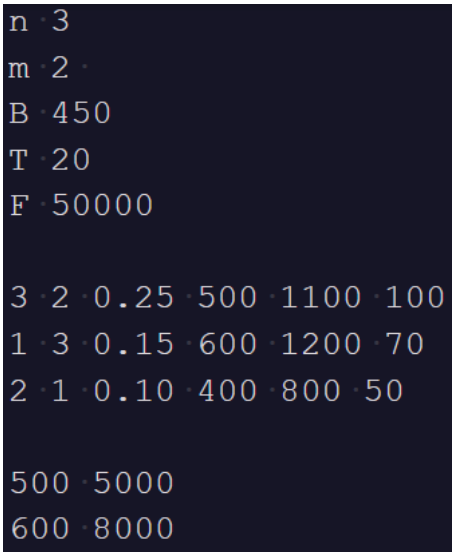
Para a entrada definimos um arquivo *.txt* em que as cinco primeiras linhas devem representar a quantidade de produtos, a quantidade de matérias-primas, a disponibilidade de horas, o tempo de troca de produto na linha e o preço fixo de produção, cada um em linha e nessa ordem. Qualquer caractere que não seja número será ignorado, porém somente um número pode ser colocado na linha.

```
n 3
m 2
B 450
T 20
F 50000
```

Em seguida é necessário pular uma linha e inserir os produtos e suas características linha por linha, com cada produto sendo uma linha. Em cada linha, separado por espaços em branco, se coloca inicialmente as matérias-primas, em ordem numérica e na quantidade definida, em seguida a quantidade de horas na linha por unidade do produto, a demanda mínima, a demanda máxima e o preço a vender por unidade do produto.

```
3 2 0.25 500 1100 100
1 3 0.15 600 1200 70
2 1 0.10 400 800 50
```

Pulando novamente uma linha é necessário adicionar as característica de cada matéria-prima (quantidade por lote e valor do lote), assim, cada linha conterá as características para cada matéria-prima, com a quantidade de matéria-prima por lote seguida do preço por lote e separadas por espaços em branco. Por fim, para os valores dados na especificação original o arquivo ficaria:



Desenvolvimento

Para a resolução desse problema, escolhemos a biblioteca OR-Tools utilizando o *MIPSolver* (*Mixed Integer Programming Solver*), principalmente pela facilidade da documentação e pela experiência obtida no projeto anterior.

Leitura do arquivo

Inicialmente foi necessário fazer a leitura do arquivo, para isso foi utilizado *regex*, podendo assim colocar em uma lista os valores numéricos (separados por espaços em branco e considerando ponto flutuante no caso da quantidade de horas por produto).

Para as cinco primeiras linhas, é pego o primeiro número encontrado nela e os valores são armazendos de acordo com a ordem estipulada. Já para os produtos e matérias-primas foram lidas linha por linha de acordo com a quantidade de produtos e matérias-primas já lidos anteriormente, colocando em uma lista multiplas outras listas com as características de cada produto, o mesmo para matéria-prima Para os produtos, a quantidade de matérias-prima de cada matérias-prima também são colocadas em uma lista.

```
[[[3, 2], 0.25, 500, 1100, 100], [[1, 3], 0.15, 600, 1200, 70], [[2, 1], 0.1, 400, 800, 50]]
```

Lista com os produtos e suas características

```
[[500, 5000], [600, 8000]]
```

Lista com as matérias-primas e suas caracterísitcas

O método final utilizado para obter os dados foi:

```
def read_data(self, filename):
      with open(filename, 'r', encoding='utf-8') as input_f:
          input_lines = input_f.readlines()

          self.number_of_products = int(re.findall(r'\b[0-9]+\b', input_lines[0])[0])
          self.number_of_materials = int(re.findall(r'\b[0-9]+\b', input_lines[1])[0])
          self.total_hours = int(re.findall(r'\b[0-9]+\b', input_lines[2])[0])
          self.time_change = int(re.findall(r'\b[0-9]+\b', input_lines[3])[0])
          self.cost_for_time = int(re.findall(r'\b[0-9]+\b', input_lines[4])[0])

          self.products = []

          for product_num in range(self.number_of_products):
              product = []
              values = re.findall(r'\b[0-9]+\.?[0-9]*\b', input_lines[6 + product_num])
```

```
                materials = []
                for material_num in range(self.number_of_materials):
                    materials.append(int(values[material_num]))

                product.append(materials)
                product.append(float(values[self.number_of_materials]))
                product.append(int(values[self.number_of_materials + 1]))
                product.append(int(values[self.number_of_materials + 2]))
                product.append(int(values[self.number_of_materials + 3]))

                self.products.append(product)

            self.materials = []

            for material_num in range(self.number_of_materials):
                self.materials.append([int(value) for value in re.findall(r'\b[0-9]+\.?[0-9]*\b', input_lines[7 + self.number_

            return [self.number_of_products, self.number_of_materials, self.total_hours, self.time_change, self.cost_for_time,
                self.products, self.materials]
```

Declaração de variáveis do Solver

Em seguida no código foi preciso definir as variáveis, essas são a quantidade de unidades de produto a serem vendido de cada produto, uma variável binária que definiria se o produto deve ser produzido ou não e a quantidade de lotes a serem comprados de cada matéria-prima. Para isso foi criada uma lista para a adição das variáveis, com as variáveis de lotes e outras listas dentro para as variáveis de cada produto (quantidade e variável binária). A ordem é da lista com as variáveis de produtos seguida das variáveis dos lotes.

```
[[n1, k1], [n2, k2], [n3, k3], w1, w2]
```

Lista com as variáveis declaradas

Assim, o código para as variáveis ficou:

```
def get_variables(self):
      self.solver = pywraplp.Solver.CreateSolver('SCIP')
      self.status = self.solver.Solve()

      infinity = self.solver.infinity()
      self.variables = []
      for product_num in range(self.number_of_products):
          self.variables.append([self.solver.IntVar(0.0, infinity, f'n{product_num + 1}'), self.solver.IntVar(0.0, 1.0, f'k

      for material_num in range(self.number_of_materials):
          self.variables.append(self.solver.IntVar(0.0, infinity, f'w{material_num + 1}'))

      print(self.variables)
```

Criação das restrições

Para as restrições, foi utilizado o método *Sum* do Solver do OR-Tools, que, a partir de uma lista, gera a soma de todos os elementos, definindo a lista da partir de *list comprehension* em *python* e definindo cada elemento pela multiplicação ou soma utilizado no somatório da restrição. Para a restrição da quantidade possível de produtos a serem vendidos foi feita uma iteração onde uma restrição de limite mínimo e máximo é criada para cada produto.

Já a restrição de horas utilizadas para produção foi feita em uma única linha a partir do somatório com as multiplicações de cada produto com a quantidade de horas por unidade dele, somado a multiplicação do somatório das variáveis binárias dos produtos com a quantidade de horas para troca de produto na linha, tudo isso sendo menor ou igual ao total de horas de produção.

A restrição da quantidade de lotes a serem comprados é feita a partir de uma iteração por cada matéria-prima, em que a lista do somatório tem como cada elemento a multiplicação entre a quantidade de matéria-prima necessária para cada produto e a variável da quantidade de unidades de produto a serem produzidos para cada produto. Esse somatório deve então ser menor que a a multiplicação da várial da quantidade de lotes da matéria-prima a serem comprados com a quantidade de matéria-prima por lote.

Por fim, o código final ficou:

```
def restriction_creations(self):
        ### Dmin and Dmax restriction
        for product_num in range(self.number_of_products):
            product_quantity = self.variables[product_num][0]
            product_exists = self.variables[product_num][1]
            dmin = self.products[product_num][2]
            dmax = self.products[product_num][3]

            self.solver.Add(product_exists*dmin <= product_quantity)
            self.solver.Add(product_exists*dmax >= product_quantity)

        ### Time restriction
        self.solver.Add(self.solver.Sum([self.variables[product][0]*self.products[product][1] for product in range(self.numb
            self.solver.Sum([self.variables[product][1] for product in range(self.number_of_products)])*self.time_change <=

        ### Number of materials restriction
        for material_num in range(self.number_of_materials):
            self.solver.Add(self.solver.Sum([self.products[product][0][material_num]*self.variables[product][0] for product
                <= self.variables[self.number_of_products + material_num]*self.materials[material_num][0])
```

Função Objetivo

Na função objetivo, foi somente utilizado o método de maximizar do *Solver* em que se utiliza os somatório com uma lista das multiplicações da variável da quantidade de unidades de cada produto a serem vendidas com o preço de cada unidade do produto, esse somatório é então subtraído do custo fixo e do somatório com as multiplicações das variáveis da quantidade de lotes de cada matéria-prima com o custo por lote da matéria-prima.

```
def objective(self):
        self.solver.Maximize(self.solver.Sum([self.variables[product][0]*\
        self.products[product][4] for product in range(self.number_of_products)]) \
        - self.cost_for_time - self.solver.Sum([self.variables[self.number_of_products + material]*\
        self.materials[material][1] for material in range(self.number_of_materials)]))
```

Resultados

Para confirmar que o resultado obedece as restrições pode-se utilizar as equações das restrições e inserir os valores obtidos pela solução do *Solver* e verificar se está dentro dos limites dados.

Fizemos testes para 3 exemplos, começando do dado pela especificação:

```
Solution:
Solution (profit) = 13760

_____________________________
Product 1:
n1 | Amount produced: 933 | Total profit: 93300.0
k1 1

_____________________________________________________
Product 2:
n2 | Amount produced: 1178 | Total profit: 82460.0
k2 1

_____________________________________________________
Product 3:
n3 | Amount produced: 0 | Total profit: 0.0
k3 0

_____________________________________________________
Raw material (lots):
w1 8

Raw material (lots):
w2 9

Time used in production: 410 | Time used in product change: 40
```

Para a restrição dos materiais temos:

$m1 : 3 * 933 + 1 * 1178 + 2 * 0 = 3977 <= 8 * 500 = 4000$

$m2 : 2 * 933 + 3 * 1178 + 1 * 0 = 5400 <= 9 * 600 = 5400$

Para a restrição dos produtos temos:

$p1 : 500 <= 933 <= 1100$

$p2 : 600 <= 1178 <= 1200$

Como k3 é zero, o produto 3 não foi produzido e portanto fica como 0.

Já para o tempo temos:

$933 * 0,25 + 1178 * 0,15 + 0 * 0,1 + (1 + 1) * 20 = 440 <= 450$

Utilizando a seguinte entrada:

```
n 4
m 2
B 450
T 20
F 50000

3 2 0.25 500 1100 100
1 3 0.15 600 1200 70
2 1 0.10 400 800 50
1 1 0.4 300 1000 150

500 5000
600 8000
```

O resultado foi:

```
Solution:
Solution (profit) = 74000
_____________________________
Product 1:
n1 | Amount produced: 0 | Total profit: 0.0
k1 0
_____________________________________________________
Product 2:
n2 | Amount produced: 0 | Total profit: 0.0
k2 0
_____________________________________________________
Product 3:
n3 | Amount produced: 0 | Total profit: 0.0
k3 0
_____________________________________________________
Product 4:
n4 | Amount produced: 1000 | Total profit: 150000.0
k4 1
_____________________________________________________
Raw material (lots):
w1 2

Raw material (lots):
w2 2

Time used in production: 400 | Time used in product change: 20
```

Para a restrição dos materiais temos:

$m1 : 3 * 0 + 1 * 0 + 2 * 0 + 1 * 1000 = 1000 <= 2 * 500 = 1000$

$m2 : 2 * 0 + 3 * 0 + 1 * 0 + 1 * 1000 = 1000 <= 2 * 600 = 1200$

Para a restrição dos produtos temos:

Como k1 é zero, o produto 1 não foi produzido e portanto fica como 0.

Como k2 é zero, o produto 2 não foi produzido e portanto fica como 0.

Como k3 é zero, o produto 3 não foi produzido e portanto fica como 0.

$p4 : 300 <= 1000 <= 1000$

Já para o tempo temos:

$0 * 0,25 + 0 * 0,15 + 0 * 0,1 + 1000 * 0,4 + (1) * 20 = 420 <= 450$

Por último temos a entrada:

```
n 4
m 3
B 500
T 15
F 40000

2 1 2 0.2 500 1100 100
1 2 1 0.35 100 1200 140
1 1 1 0.10 400 800 50
4 5 3 0.4 100 1000 300

500 5000
600 8000
500 6000
```

Cujo resultado foi:

```
Solution:
Solution (profit) = 129000

_____________________________
Product 1:
n1 | Amount produced: 0 | Total profit: 0.0
k1 0

_____________________________________________________
Product 2:
n2 | Amount produced: 200 | Total profit: 28000.0
k2 1

_____________________________________________________
Product 3:
n3 | Amount produced: 0 | Total profit: 0.0
k3 0

_____________________________________________________
Product 4:
n4 | Amount produced: 1000 | Total profit: 300000.0
k4 1

_____________________________________________________
Raw material (lots):
w1 9

Raw material (lots):
w2 9

Raw material (lots):
w3 7

Time used in production: 470 | Time used in product change: 30
```

Para a restrição dos materiais temos:

$m1 : 2 * 0 + 1 * 200 + 1 * 0 + 4 * 1000 = 4200 <= 9 * 500 = 4500$

$m2 : 1 * 0 + 2 * 200 + 1 * 0 + 5 * 1000 = 5400 <= 9 * 600 = 5400$

$m3 : 2 * 0 + 1 * 200 + 1 * 0 + 3 * 1000 = 3200 <= 9 * 500 = 4500$

Para a restrição dos produtos temos:

Como k1 é zero, o produto 1 não foi produzido e portanto fica como 0.

$p2 : 100 <= 200 <= 1200$

Como k3 é zero, o produto 3 não foi produzido e portanto fica como 0.

$p4 : 300 <= 1000 <= 1000$

Já para o tempo temos:

$0 * 0,2 + 200 * 0,35 + 0 * 0,1 + 1000 * 0,4 + (1 + 1) * 15 = 500 <= 500$